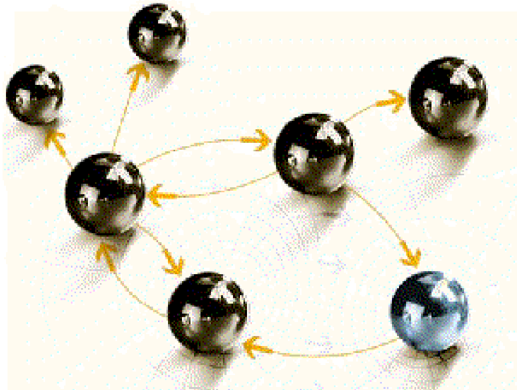


cps4it

Grundlagen der Programmierung

cps4it

consulting, projektmanagement und seminare für die informationstechnologie

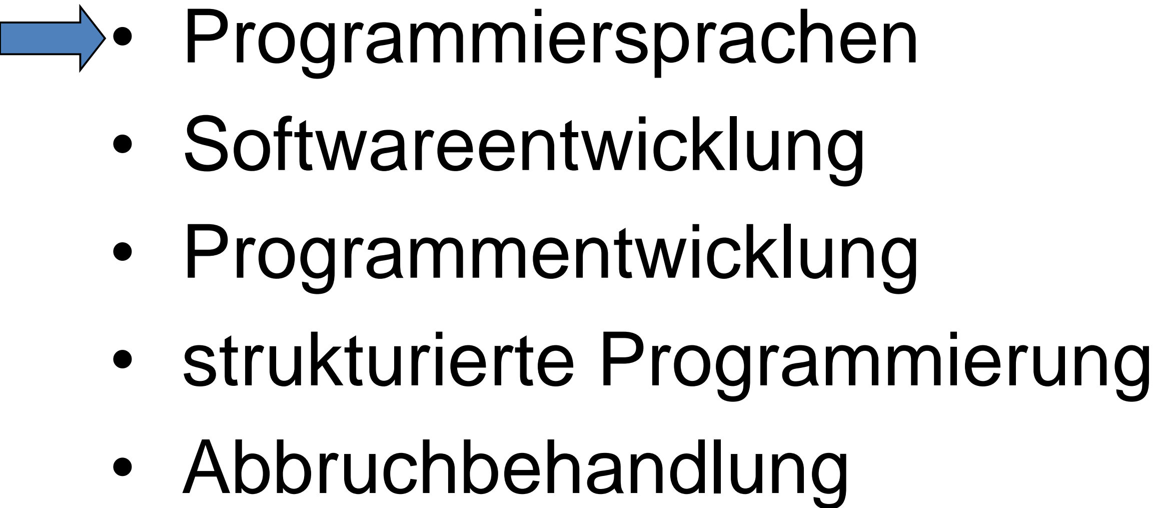
Ralf Seidler, Stromberger Straße 36A, 55411 Bingen

Fon: +49-6721-992611, Fax: +49-6721-992613, Mail: ralf.seidler@cps4it.de

Internet: http://www.cps4it.de

Inhalt

cps4it



- Programmiersprachen
- Softwareentwicklung
- Programmentwicklung
- strukturierte Programmierung
- Abbruchbehandlung

Programmiersprachen

cps4it

Begriffe

Definiton

ASM

JAVA

Generation

Stile

Umwan-
deln

Programmiersprachen

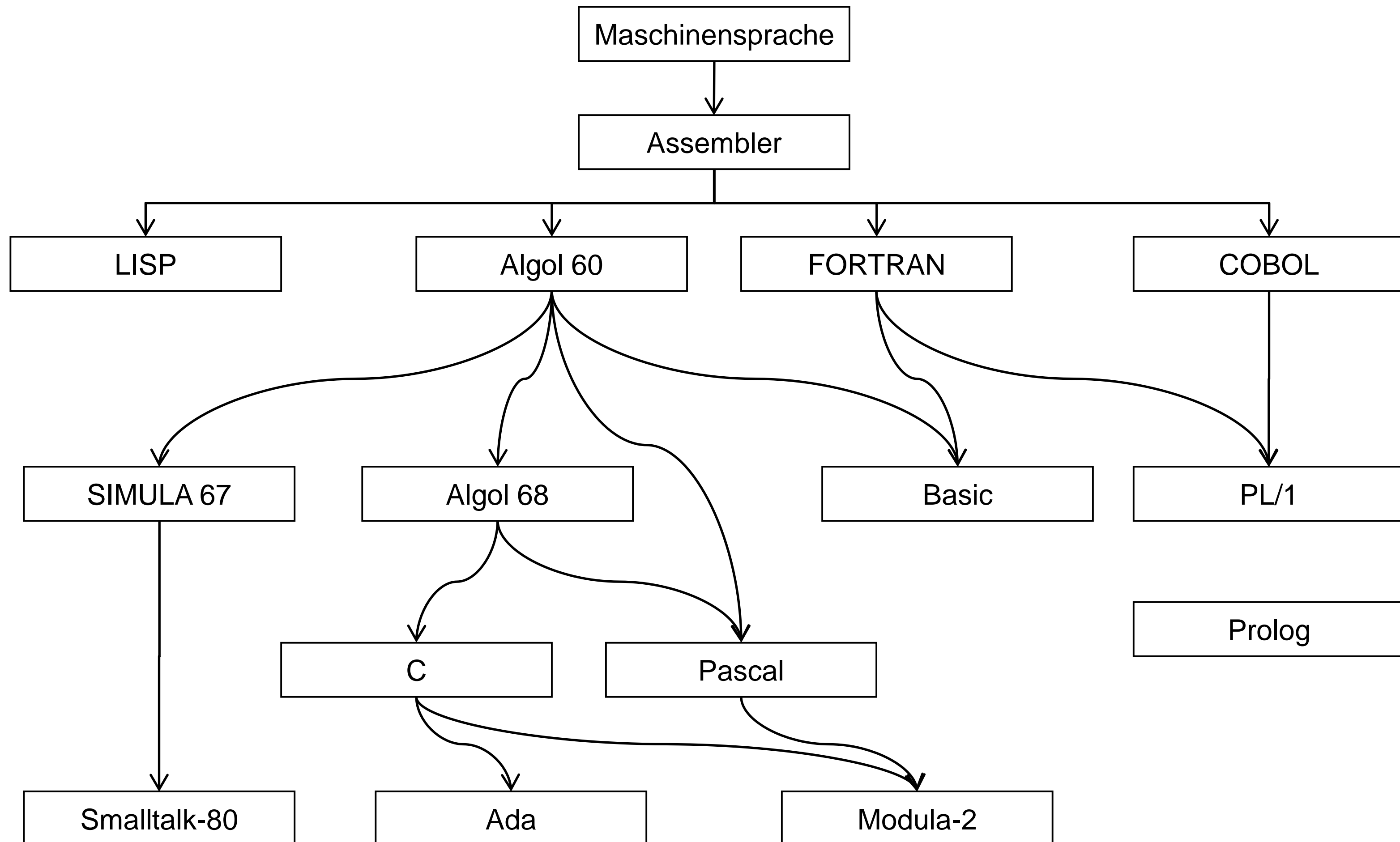
cps4it

Historie

- 1. Generation – Maschinensprache
- 2. Generation – Assembler
- 3. Generation
  - C, COBOL, PASCAL, FORTRAN, PL1, ALGOL, ADA, Basic, SIMULA, JAVA etc.
- 4. Generation
  - Natural (4GL), SQL, VBA, Office-Makros
- 4.5. Generation
  - OO-Sprachen wie JAVA, COBOL, SmallTalk
- 5. Generation – PROLOG (VHLL)

Programmiersprachen

Verwandtschaft – nicht vollständig



Programmiersprachen

cps4it

Ausblick – 1/3 (persönliche Auffassung)

- Maschinensprache
  - abhängig von Hardware
  - trotzdem auch künftig notwendig
  - z.B. Chips
- ASM
  - nicht unabhängig von Hardware
  - schwierige Wartung, da kein Nachwuchs
  - wird wohl auslaufen -> C bzw. Turbo PASCAL
  - wird wohl nicht auslaufen wegen Stärke der IBM ??

Programmiersprachen

cps4it

Ausblick – 2/3 (persönliche Auffassung)

- 3. Generation
  - COBOL/PL1 mit hohem Marktanteil; wird daher so schnell nicht wegfallen
  - C übernimmt wohl ASM-Funktionen (nur Teile)
    - Versuch, unabhängig von Hardware zu sein, was nicht komplett gelingt
  - JAVA hat Chancen, aber
    - Eine echte Plattformunabhängigkeit gibt es nicht.
    - Wer versucht, alles mit JAVA zu lösen, wird scheitern.
  - alle anderen sind abhängig von starken Personen

Programmiersprachen

cps4it

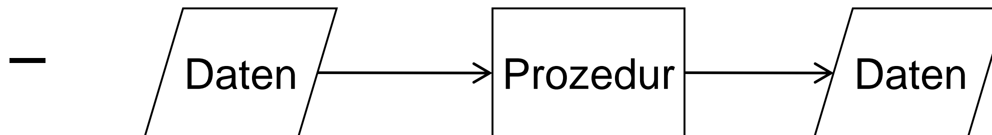
Ausblick – 3/3 (persönliche Auffassung)

- 4. Generation
  - DB-bezogene Sprachen bleiben
- 4.5. Generation
  - OO-Sprachen
    - SmallTalk gibt es (fast) nicht mehr
    - JAVA hat Chancen
    - COBOL als OO-Sprache ohne Zukunft
    - Problem: Overhead
- 5. Generation
  - d.h. KI, die immer noch nicht (Massen-)Markt fähig ist (Fußball-Roboter, Staubsauber-Roboter, Autos ohne Fahrer etc.)

Programmiersprachen

cps4it

Programmierstile – 1/4

- imperativ
  - Wertzuweisungen
  - Verzweigungen
  - Basis: Neumann-Rechnerarchitektur
- prozedural
  - Iteration
  - Selektion
  - 

Programmiersprachen

cps4it

Programmierstile – 2/4

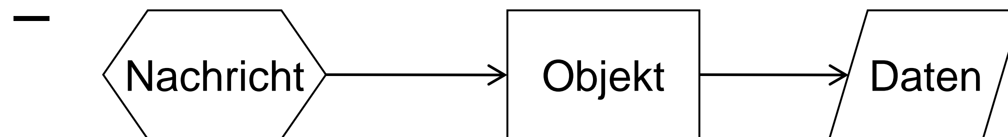
- funktional
  - einfache Funktionen zu komplexen zusammen bauen
  - Rekursion
  - Variablen präsentieren Wert und sind daran gebunden oder eben ungebunden
- logisch
  - es werden nur Fakten und Eigenschaften (Regeln) des Problems formuliert
  - Formulierung über Formalismus
  - System sucht eigenständig die Lösung
  - Variablenkonzept ähnlich wie bei funktionalem Stil

Programmiersprachen

cps4it

Programmierstile – 3/4

- Objekt-orientiert
  - die Welt ist eine Menge von Objekten
  - Objekte kommunizieren über Nachrichten
  - Objekt ist beschrieben durch Daten und Methoden
  - Daten charakterisieren Attribute des Objekts
  - Methoden regeln Zugriff auf Daten
  - streng hierarchisch orientiert
  - Vererbung
  -



Programmiersprachen

cps4it

Programmierstile – 4/4

- Parallelisierung
  - Prozesse können parallel abgearbeitet werden
  - es stehen mehrere Prozessoren zur Verfügung
  - Ergebnisse werden zusammen geführt
  - zwei Grenzwerte
    - natürliche Parallelität auf Einprozessorrechnern, die sequentiell abgearbeitet werden
    - sequentielle Prozesse werden für Mehrprozessorrechner parallel abgebildet
- Synchronisierung

Programmiersprachen

cps4it

Programmierstile – und die Sprachen

- Es gibt (nahezu) keine reine Sprache
  - OO: Smalltalk, Eiffel, andere mit OO-Erweiterung
  - prozedural: COBOL, PL1, REXX etc.
  - funktional: LISP
  - logisch: PROLOG
  - parallele Systeme: SIMULA, MODULA, ADA

Programmiersprachen

cps4it

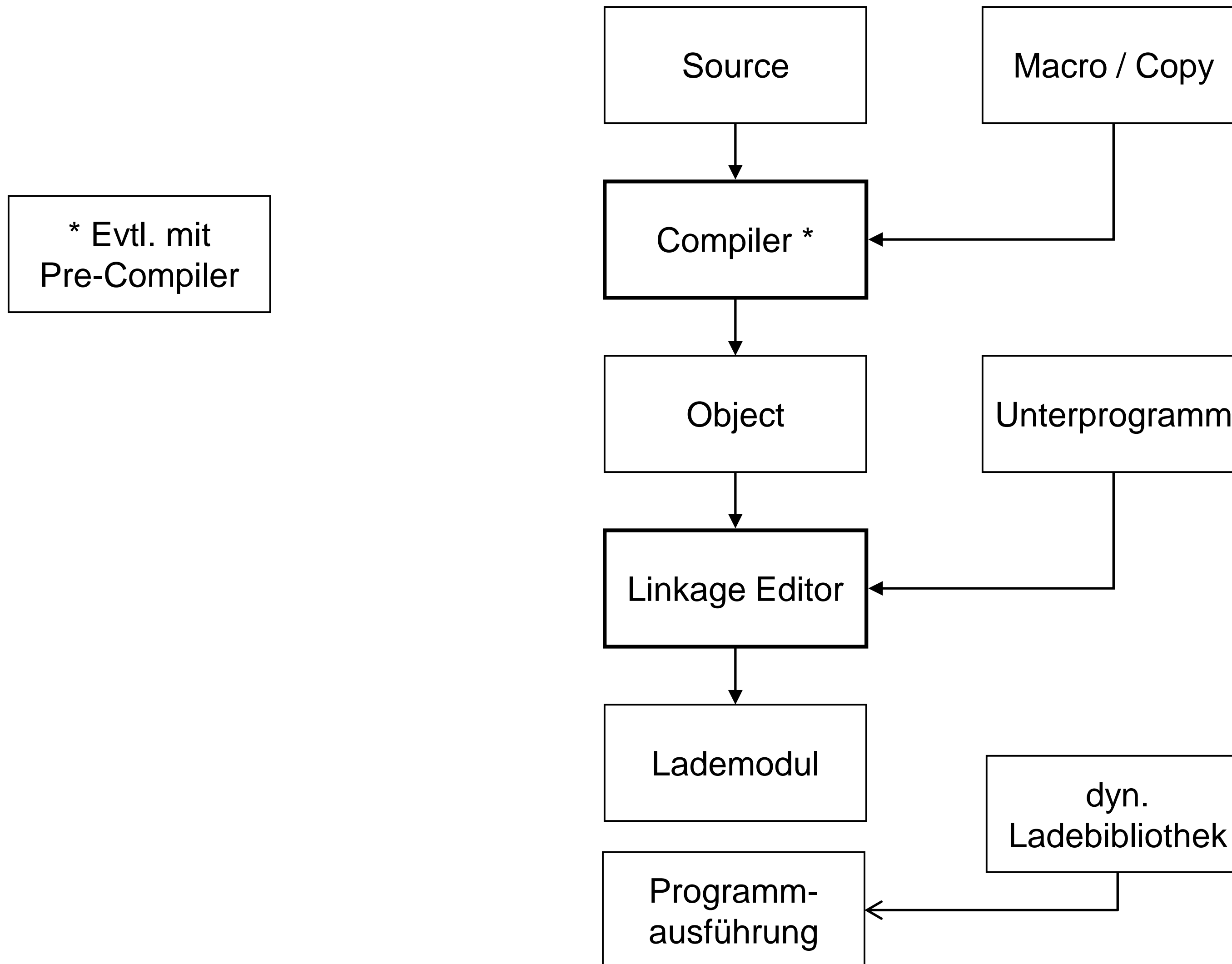
Arbeitsschritte beim Implementieren

- Editieren
- Übersetzen
- Binden
- Laden
- Ausführen

Programmiersprachen

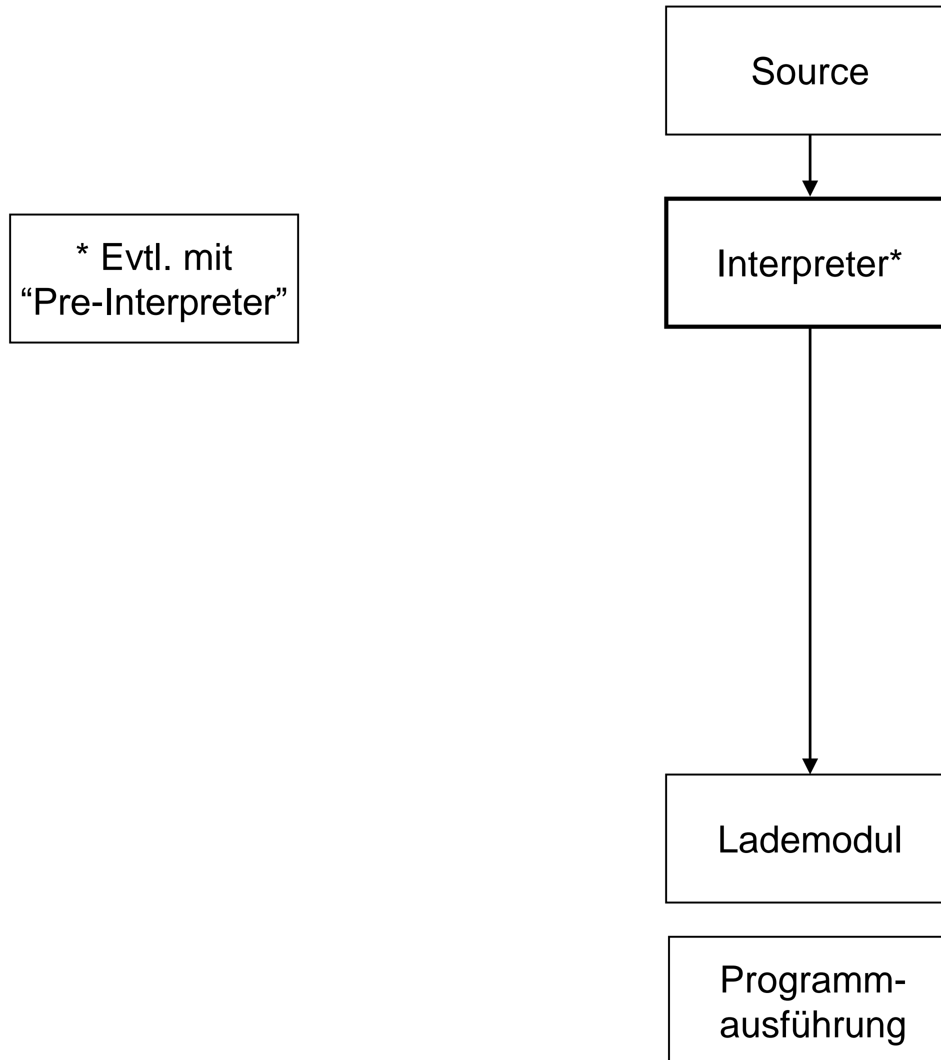
cps4it

Arbeitsschritte beim Implementieren – Schaubild mit Compiler



Programmiersprachen

Arbeitsschritte beim Implementieren – Schaubild mit Interpreter



Programmiersprachen

Zwischenformen beim Übersetzen

- Compreter
  - Zwischencode (P-Code oder Byte-Code)
  - Zwischencode soll unabhängig von Hardware sein
  - Zwischencode wird mit Interpreter ausgeführt
- aktuell
  - Compile auf beliebigem Rechner mit erzeugen von Byte-Code
  - Bilden ausführbares Modul auf Zielmaschine

Programmiersprachen

cps4it

Aufgabe des Binders und der Laufzeitumgebung

- Binder
  - Zusammenfügen aller benötigten Ressourcen wie
    - Referenzen von Dateien
    - Unterprogramme bzw. Referenzen dazu
  - erzeugen von Maschinencode
- Laufzeitumgebung
  - Laden des Programms
  - Zuordnen aller externen Referenzen mit absoluten Adressen
  - Kontrolle bei der Ausführung

Inhalt

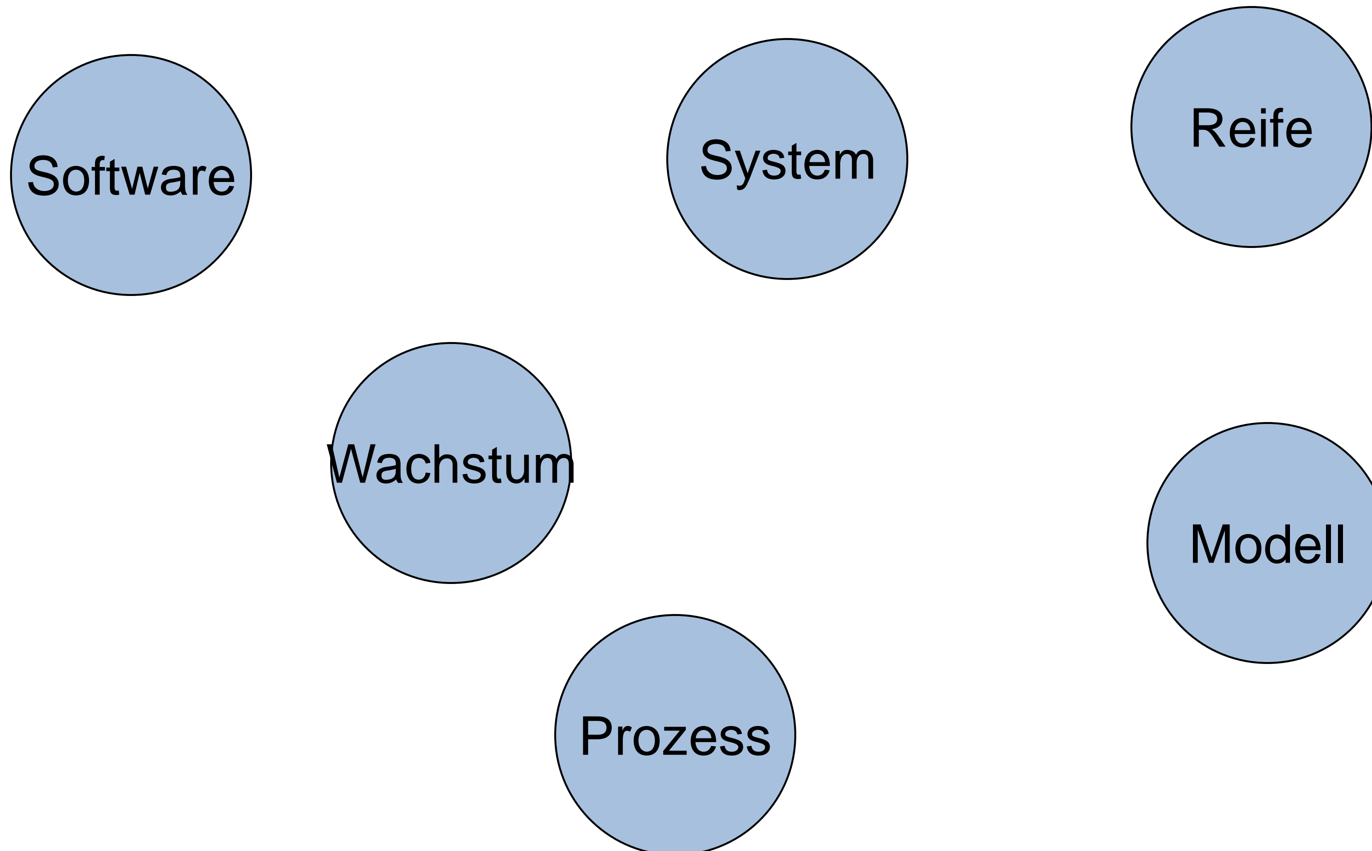
cps4it

- Programmiersprachen
- Softwareentwicklung
- Programmentwicklung
- strukturierte Programmierung
- Abbruchbehandlung

Softwareentwicklung

cps4it

Begriffe



Definition von Software

- Software ist ein Produkt. Es ist eine Kombination von Programmen, Dokumentationen und Daten. Sie unterliegt dem allgemeinen Zyklus von Anwendungssystemen.

Softwareentwicklung

cps4it

Lebenszyklus – 1/2

- Systementwicklung
  - Entwicklung der Software
- Systemeinführung
  - Vorbereitungen zur Nutzung der Software
- Wachstum
  - Verbreitung der Nutzung der Software
- Reife
  - Umsetzen von Verbesserungen an der Software, Beseitigung von Fehlern

Softwareentwicklung

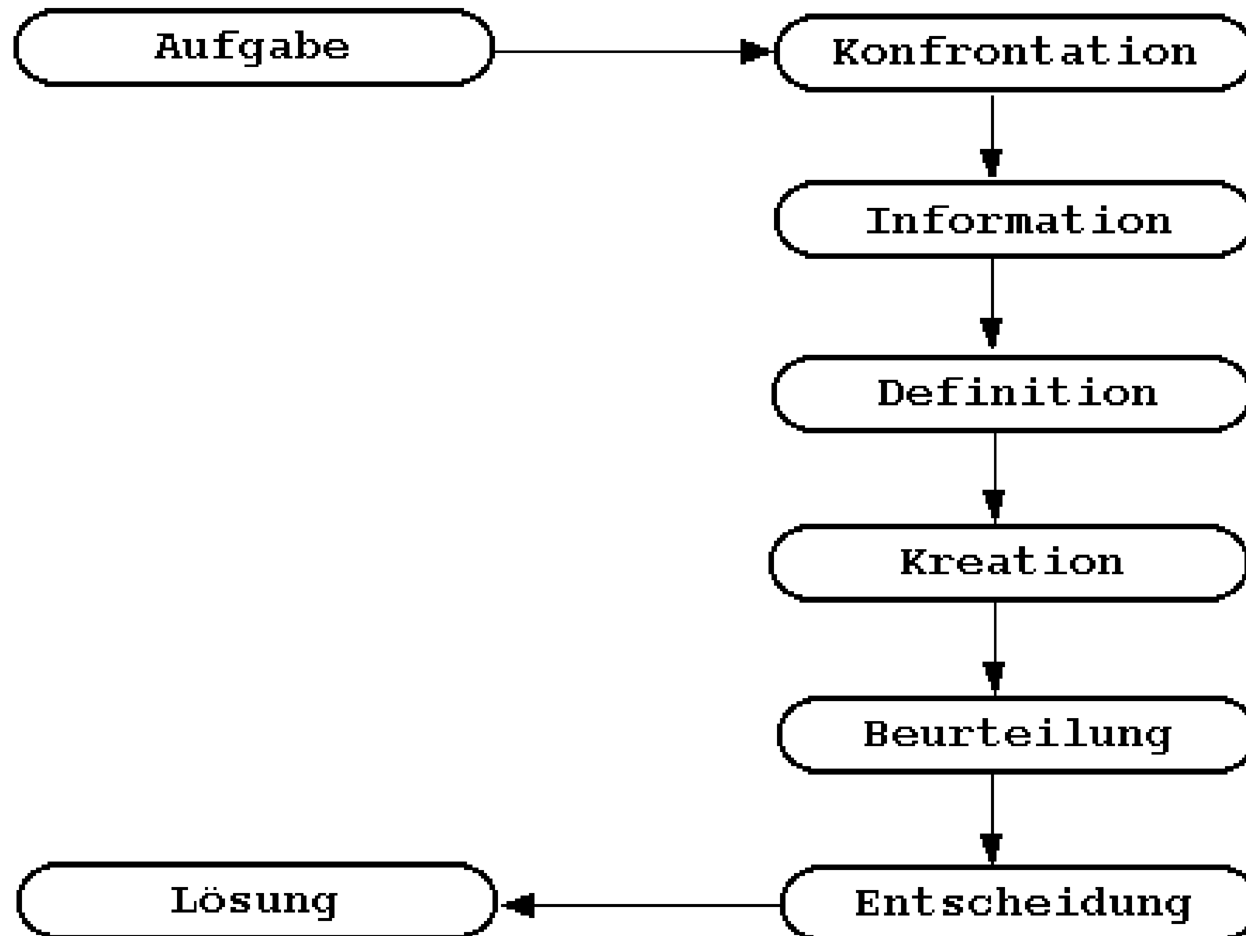
cps4it

Lebenszyklus – 2/2

- Rückgang
  - Schrittweiser Übergang zu einem neuen Software-Produkt oder zu einer neuen Version des Produkts
  - Ablösung des Produkts mit Löschen aller nicht mehr benötigten Ressourcen / Objekten

Softwareentwicklung

Problemlösungsprozess

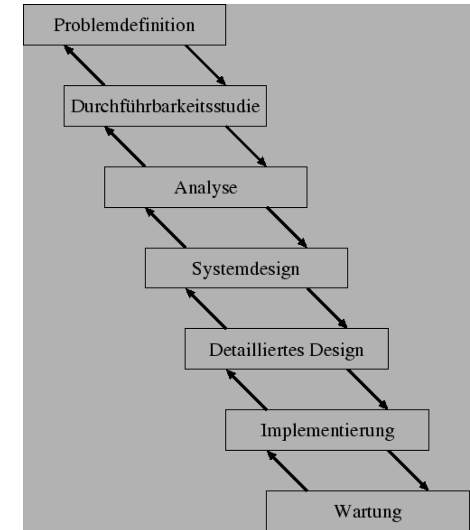


Softwareentwicklung

cps4it

Wasserfallmodell – Phasen

- Planung / Grobentwurf
- Analyse / Fachentwurf
- Design / technischer Entwurf
- Programmierung mit Modultest
- Integration und Systemtest
- Auslieferung, Einsatz und Wartung

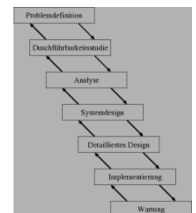


Softwareentwicklung

Wasserfallmodell – Schwachstellen

- tatsächlicher Prozess nicht nacheinander
- Testen und damit Einbinden der Fachbereiche sehr spät
- Prototyping nicht möglich

- Beginn der Entwicklung von verschiedenen Modellen, die den Entwicklungsprozess variabler berücksichtigen



cps4it

Entwicklungsprozess – Nutzen von Modellen

- Planungssicherheit (vergleichbare Produkte vergleichbar erstellen)
- Effizienz (nichts vergessen, keine Nachbereinigungen)
- Kostenreduzierung (Fehler schnell erkennen)
- Qualität (von Anfang an kontrolliert)

Softwareentwicklung

cps4it

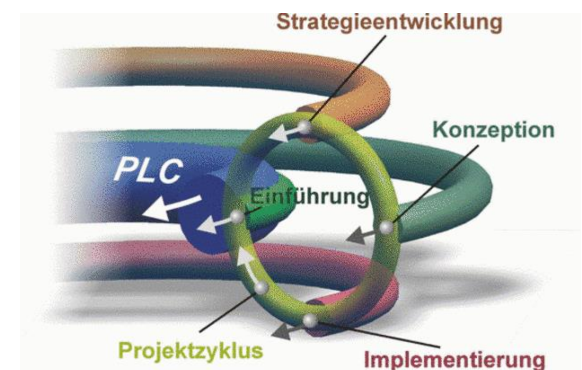
Entwicklungsprozess – Methoden

- Wasserfallmodell (1970)
- Iteratives / evolutionäres Prototyping (1980)
- V-Modell (1986)
- Spiralmodell (1988)
- agile Softwareentwicklung (1990)
- extreme Programmierung (1999)
- V-Modell XT (2005)
- agiles V-Modell (aktuell)

Softwareentwicklung

Iteratives Modell – Eigenschaften

- Das Softwareprodukt wird allmählich und stufenweise entwickelt.
- Die Steuerung basiert aufgrund gesammelter Erfahrungen bei der Anwendung.
- Die Wartung der Software wird als Erstellung einer neuen Version verstanden.



Softwareentwicklung

cps4it

Iterratives Modell – Vorteile / Nachteile

- Gut, wenn der Auftraggeber der Softwareentwicklung seine Anforderungen nicht vollständig überblickt.
- Gut, da die Entwicklung sich auf lauffähige Teillösungen konzentriert, die über verschiedene Versionen nacheinander freigegeben werden.
- Gefahr, dass zu einem späteren Entwicklungszeitpunkt die Architektur geändert und angepasst werden muss.

Softwareentwicklung

V-Modell

- System-Anforderungsanalyse
- System-Entwurf
- SW-/HW-Anforderungsanalyse
- SW-Grobentwurf
- SW-Feinentwurf
- SW-Implementierung
- SW-Integration
- System-Integration
- Überleitung in die Nutzung

cps4it

Spiralmodell – Eigenschaften

- Iterativer Prozess, jeder Zyklus enthält:
  - Festlegung von Zielen, Alternativen und Rahmenbedingungen
  - Evaluierung der Alternativen und das Erkennen und Reduzieren von Risiken
  - Realisierung und Überprüfung des Zwischenprodukts
  - Planung der Projektfortsetzung.
- Die Phasen des Wasserfallmodells werden mehrfach spiralförmig durchlaufen.

Softwareentwicklung

Spiralmodell – Grafik

Kosten

Fortschritte

Festlegen der Ziele, Beurteilen von Alternativen

Risikoanalyse

Zustimmung durch Überprüfung

Lebens-zyklus-planung

Risiko-analyse

Risiko-analyse

Risiko-analyse

Prototyp 1

Prototyp 2

betriebs-fähiger Prototyp

Konzept für Betrieb

Planung der Anforder-ungen

Anfor-derun-gen

Grob-ent-wurf

Fein-ent-wurf

Entwicklungs-plan

Verifikation & Validation

Code

Testplanung

Verifikation & Validation

Integration

Planung des nächsten Zyklus

Abnahme

Implemen-tierung

Test

Entwicklung und Test

Softwareentwicklung

cps4it

agile Softwareentwicklung – 1/2

- Eigenschaften
  - reine Entwurfsphase auf ein Mindestmaß reduzieren
  - im Entwicklungsprozess so früh wie möglich zu ausführbarer Software gelangen
  - Abstimmung jederzeit mit Kunden
- Ziele
  - flexible Handhabung von Kundenwünschen
  - hohe Kundenzufriedenheit
  - einfach (KISS)
  - gemeinsamer Code-Besitz
  - vorhandene Ressourcen mehrfach verwenden

Softwareentwicklung

cps4it

agile Softwareentwicklung – 2/2

- Beispiele
  - Adaptive Software Development (ASD)
  - Dynamic System Development Method (DSDM)
  - Extreme Programming (XP)
  - Feature Driven Development (FDD)
  - Pragmatic Programming
  - Scrum
  - Software-Expedition / Universal Application
  - Usability Driven Development (UDD)
  - Testgetriebene Entwicklung
- http://www.agilemanifesto.org/principles.html

Softwareentwicklung

extreme Programmierung

- Beispiel eines Teams bei DaimlerChrysler
- kleine bis mittelgroße Teams
- „übertriebener“ Einsatz anerkannter Techniken
- 14 Grundprinzipien

- interessant: es gibt keine (Uni-) Web-Adresse mehr mit Vorlesungsfolien / kaum Artikel

Softwareentwicklung

cps4it

Gesetze von Murphy

1. Die Dinge sind komplexer als sie scheinen!
2. Die Dinge brauchen länger als erwartet!
3. Die Dinge kosten mehr als vorgesehen!
4. Wenn etwas schief gehen kann,
   so geschieht es!

Anmerkung: Murphy war ein Optimist !

Softwarequalität – 1/2

- Software muss zuverlässig die Aufgaben lösen, zu deren Erledigung sie entwickelt wurde. Kann die Software (unter bestimmten Umständen) eine Teilaufgabe nicht (zuverlässig) lösen, so sollte sie dies dem Nutzer - unmissverständlich - mitteilen
- Sie muss so einfach wie möglich zu bedienen sein, d.h. sie muss benutzerfreundlich sein.
  - Achtung: das Auslösen kritischer Operationen darf nicht zu einfach sein!

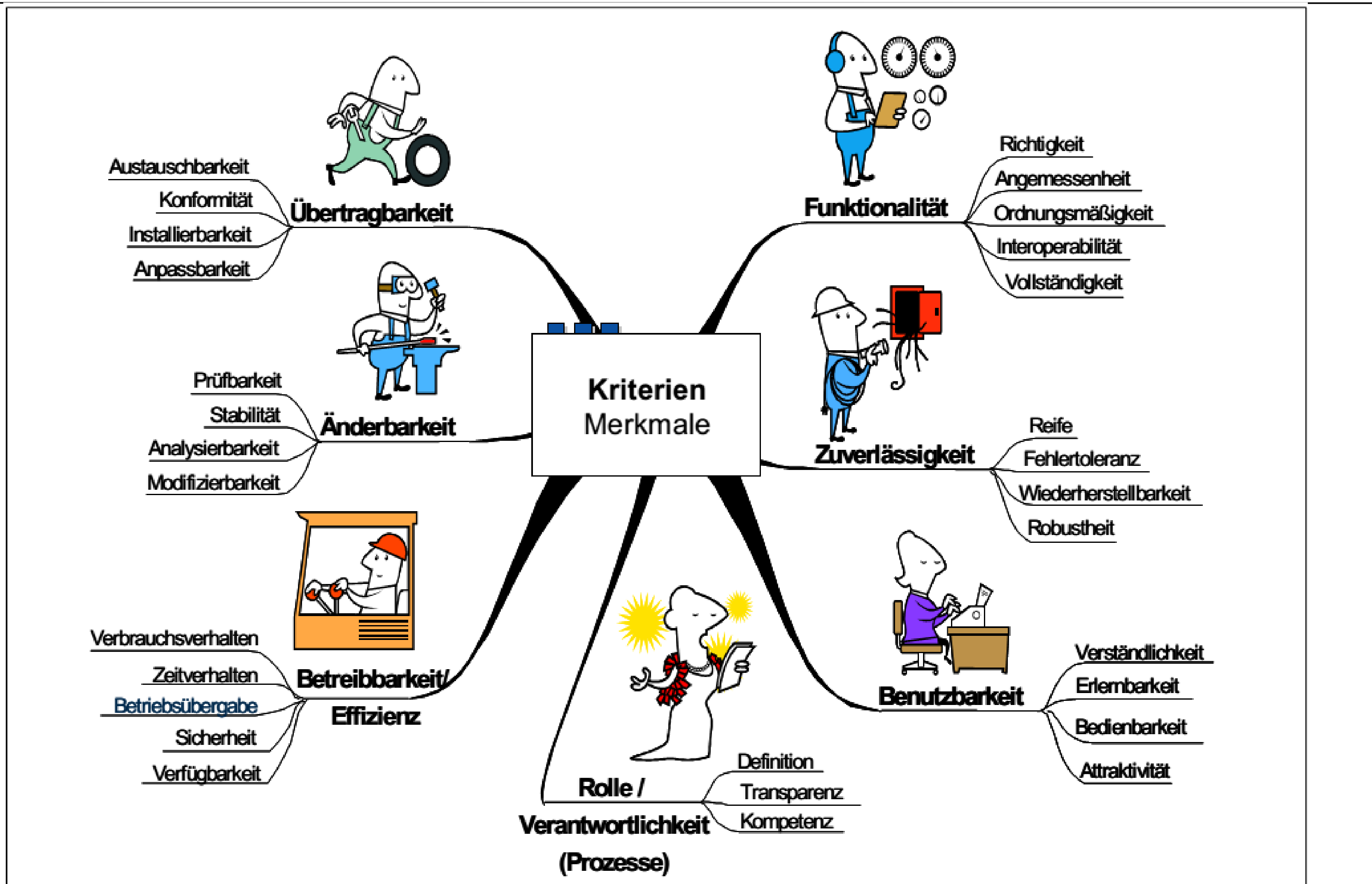
Softwarequalität – 2/2

- Software muss so einfach wie möglich wartbar sein; gute Wartbarkeit ist Voraussetzung für Flexibilität !
- Software sollte so effektiv wie notwendig arbeiten
  - Achtung: zu uneffektiv arbeitende Software ist benutzerunfreundlich!

Softwareentwicklung

cps4it

Softwarequalität – eine andere Sicht



Softwareentwicklung

cps4it

Softwarequalität – Übertragbarkeit – Definition

- Wie leicht lässt sich das Objekt in eine andere Umgebung übertragen?
- Eignung des Objektes, von der Umgebung in eine andere übertragen werden zu können. Umgebung kann organisatorische, Hardware- oder Software-Umgebung sein

Softwarequalität – Übertragbarkeit – Begriffe – 1/2

- Austauschbarkeit
  - Möglichkeit, das Objekt anstelle eines spezifizierten anderen Objektes zu verwenden, sowie der dafür notwendige Aufwand (-> klar definierte Schnittstellen (Daten Input/Output); klar abgegrenzte fachliche Funktionen in technisch gekapselten Einheiten)
- Konformität
  - Grad, in dem das Objekt Normen und Vereinbarungen erfüllt (R+V Namenskonventionen, Programmierrichtlinien, Style-Guides, ...)

cps4it

Softwarequalität – Übertragbarkeit – Begriffe – 2/2

- Installierbarkeit
  - Aufwand, der zum installieren des Objektes in einer festgelegten Umgebung notwendig ist
    (-> Standardisierung in Verfahren, technische Unterstützung, Hilfen, ...)
- Anpassbarkeit
  - Fähigkeit des Objektes, diese an verschiedene Umgebungen anzupassen
    (-> z.B. auch durch 3-Schichten-Architektur, systemunabhängige Programmierung, ...)

cps4it

Softwarequalität – Änderbarkeit – Definition

- Welchen Aufwand erfordert die Durchführung vorgegebener Änderungen am Objekt?
- Aufwand, der zur Durchführung vorgegebener Änderungen notwendig ist. Änderungen können Korrekturen, Verbesserungen oder Anpassungen an Änderungen der Umgebung, der Anforderungen oder der funktionalen Spezifikation einschließen.

Softwarequalität – Änderbarkeit – Begriffe – 1/3

- Prüfbarkeit
  - Aufwand, der zur Prüfung des geänderten Objektes notwendig ist (Möglichkeiten von Regressionstests, Komplexität der Anwendung im Sinne von möglichem Abdeckungsgrad bei der Prüfung)
- Stabilität
  - Wahrscheinlichkeit des Auftretens unerwarteter Wirkung von Änderungen (Kapselung, eingrenzbarer Wirkungsbereich von Änderungen)
  - siehe Programmierrichtlinien objekt-orientierter Sprachen

cps4it

Softwarequalität – Änderbarkeit – Begriffe – 2/3

- Analysierbarkeit
  - Aufwand, um Mängel oder Ursachen von Versagen zu diagnostizieren oder um änderungsbedürftige Teile zu bestimmen (Debug-Möglichkeit, aber Nachvollziehbarkeit, technische Dokumentation, Grad der Vernetzung im System, Möglichkeit zur Analyse von Schwachstellen (z.B. Fehlerhäufigkeit in bestimmten Teilbereichen))
  - siehe Programmierrichtlinien

cps4it

Softwarequalität – Änderbarkeit – Begriffe – 3/3

- Modifizierbarkeit
  - Aufwand zur Ausführung von Verbesserungen, zur Fehlerbeseitigung, aber auch Anpassung an Umgebungsänderungen (Analyseaufwand, finden der Stellen, die geändert werden müssen, Vollständigkeit der Änderung, ...)

Softwareentwicklung

cps4it

Softwarequalität – Betreibbarkeit / Effizienz – Definition

- Wie liegt das Verhältnis zwischen Leistungsniveau des Objekts und den eingesetzten Betriebsmitteln?
- Verhältnis zwischen Leistungsniveau und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen; Sicherstellung der Betreibbarkeit im Rahmen geltender Richtlinien

cps4it

Softwarequalität – Betreibbarkeit / Effizienz – Begriffe – 1/2

- Verbrauchsverhalten
  - Anzahl und Dauer der benötigten Betriebsmittel bei der Erfüllung der Funktionen. Ressourcenverbrauch, CPU-, Festplattenzugriffe , usw.
- Zeitverhalten
  - Antwort- und Verarbeitungszeiten sowie Durchsatz bei der Funktionsausführung
- Verfügbarkeit
  - Zielsetzung, in welchem Maße ein Objekt / Dienstleistung (im Sinne Projektergebnis) im Betrieb zur Verfügung stehen muss (Bsp . 7x24 h)

Softwarequalität – Betreibbarkeit / Effizienz – Begriffe – 2/2

- Betriebsübergabe
  - Alle notwendigen Dokumente, Know-how-Transfers und Abstimmungen zur Übergabe des Produktes in den Betrieb liegen vor / sind getroffen
- Sicherheit
  - Fähigkeit, unberechtigten Zugriff, sowohl versehentlich als auch vorsätzlich, auf Programme und Daten zu verhindern; aber auch die Möglichkeit der Vorgabe von unterschiedlichen Berechtigungsstufen bis hin zur Unterstützung des 4-Augen-Prinzips

Softwarequalität – Funktionalität – Definition

- Inwieweit besitzt das Objekt die geforderten Funktionen?
- Vorhandensein von Funktionen mit festgelegten Eigenschaften. Diese Funktionen erfüllen die definierten Aufgaben

cps4it

Softwarequalität – Funktionalität – Begriffe – 1/3

- Richtigkeit
  - Liefern der richtigen Vereinbarungen oder Wirkungen, z.B. die benötigte Genauigkeit von berechneten Werten
- Angemessenheit
  - Eignung von Funktionen für spezifizierte Aufgaben, z.B. Aufgaben-orientierte Zusammensetzung von Funktionen aus Teilfunktionen

Softwarequalität – Funktionalität – Begriffe – 2/3

- Ordnungsmäßigkeit
  - Nachvollziehbarkeit der Ausführung der Funktionen und Unterstützung zur Kontrolle der Funktionsnutzung
  - Möglichkeit zur Aussteuerung von Vorgängen für Stichprobenkontrollen
  - Dokumentation der Funktionalität, Nachvollziehbarkeit von Änderungen in der Dokumentation, Release Management
  - Historisierung in Bezug auf Entwicklungschronologie

Softwareentwicklung

cps4it

Softwarequalität – Funktionalität – Begriffe – 3/3

- Interoperabilität
  - Fähigkeit, mit vorgegebenen Systemen zusammen zu wirken
  - Schnittstellenkonformität
  - Einbettung in die firmenspezifische Systemlandschaft
  - Anbindung und Nutzung der Zentralsysteme
- Vollständigkeit
  - Funktionalität ist durchgängig und vollständig in Bezug auf die (maschinelle) Unterstützung eines bestimmten Prozesses / Arbeitsablaufs

cps4it

Softwarequalität – Zuverlässigkeit – Definition

- Kann das Objekt ein bestimmtes Leistungsniveau unter bestimmten Bedingungen über einen bestimmten Zeitraum aufrecht erhalten?

Softwareentwicklung

cps4it

Softwarequalität – Zuverlässigkeit – Begriffe – 1/2

- Reife
  - Geringe Versagenshäufigkeit durch Fehlerzustände
  - gut getestete Anwendung ohne Fehler; Fehler meint hierbei "echte" Fehler, nicht Fehler in der Bedienung
- Fehlertoleranz
  - Fähigkeit, ein spezifiziertes Leistungsniveau bei Fehlern oder Nicht-Einhaltung ihrer spezifizierten Schnittstelle zu bewahren
  - Reaktion des Systems auf Fehler aus z.B. Schnittstellenfehlern oder Eingabefehlern

Softwareentwicklung

cps4it

Softwarequalität – Zuverlässigkeit – Begriffe – 2/2

- Wiederherstellbarkeit
  - Fähigkeit, bei einem Versagen das Leistungsniveau wiederherzustellen und die direkt betroffenen Daten wieder zu gewinnen. Zu berücksichtigen sind dafür die betroffene Zeit und der betroffene Aufwand.
- Robustheit
  - Fähigkeit, ein stabiles System bei Eingaben zu gewährleisten, die gar nicht vorgesehen sind.
  - Das Objekt hält DAU-Tests stand.

Softwarequalität – Benutzbarkeit – Definition

- Welchen Aufwand fordert der Einsatz des Objektes von den Benutzern und wie wird er von diesen beurteilt?
- Aufwand, der zur Benutzung erforderlich ist, und individuelle Beurteilung der Benutzung durch eine festgelegte oder vorausgesetzte Benutzergruppe

cps4it

Softwarequalität – Benutzbarkeit – Begriffe – 1/2

- Verständlichkeit
  - Aufwand für den Benutzer, das Konzept und die Anwendung zu verstehen; Grad in dem Oberflächen den Bildschirmrichtlinien und evtl. spezifischen Style-Guides entsprechen
- Erlernbarkeit
  - Aufwand für den Benutzer, die Anwendung zu erlernen
  - z.B. Bedienung, Ein- und Ausgabe, selbsterklärende Anwendungen, Unterstützung durch Hilfefunktionen

cps4it

Softwarequalität – Benutzbarkeit – Begriffe – 2/2

- Bedienbarkeit
  - Aufwand für den Benutzer, die Anwendung zu bedienen
  - z.B. Menuführung, Kontext-bezogene Selektionsboxen und Hilfen, Abwägung von Massenbearbeitungen versus Spezialgeschäft)
- Attraktivität
  - Anziehungskraft der Anwendung gegenüber dem Benutzer

Softwarequalität – Rolle / Verantwortlichkeit – Definition

- Inwieweit besitzen die Personen den notwendigen Skill, um im Sinne des Ergebnisses richtig und angemessen zu arbeiten?

Softwarequalität – Rolle / Verantwortlichkeit – Begriffe

- Definition
  - Klare Definition und Abgrenzung der Rolle, Zielsetzung
- Transparenz
  - Kennt alle Beteiligten / Betroffenen die Beschreibung / Zielsetzung
- Kompetenz
  - Ausbildung, Erfahrung, Skill / Fachwissen, persönliche Kompetenz

cps4it

Softwarequalität – Strukturierung – 1

- Allgemeiner Nutzen
  - Brauchbarkeit
    - Portabilität
    - Zuverlässigkeit
    - Effizienz
    - Benutzerfreundlichkeit
  - Wartbarkeit
    - Testbarkeit
    - Verständlichkeit
    - Änderbarkeit

Softwareentwicklung

cps4it

Softwarequalität – Strukturierung – 2

- Geräteunabhängigkeit
- Autarkie
- Genauigkeit
- Vollständigkeit
- Robustheit
- Integrität
- Konsistenz
- Zählbarkeit
- Geräte-Effizienz
- Zugänglichkeit
- Assimilationsfähigkeit
- Selbsterklärung
- Strukturierung
- Kompaktheit
- Lesbarkeit
- Erweiterbarkeit

Grundsätze für die Softwareentwicklung – 1

- Trauen Sie Ihren Nutzern zu, dass sie in der Lage sind, jede sich bietende Fehlermöglichkeit zu nutzen.
  (Murphy'sche Regel: Wenn etwas schief gehen kann, so geschieht es!)
- Gehen Sie als Software-Entwickler vom schlimmsten Fall aus:
  – Sie müssen ihr eigenes Produkt nutzen.
  – Sie müssen ihr eigenes Produkt warten.
- Die Produktivität des Nutzers ist zu messen an der Anzahl der Eingabehandlungen, die er tätigen muss, bis er das Problem gelöst hat.

Softwareentwicklung

Grundsätze für die Softwareentwicklung – 2

- Routinierte Nutzer bewerten die Benutzeroberfläche eines Programms anders als neue oder gelegentliche Nutzer.
  Der routinierte Nutzer sieht mehr den Aufwand bei der täglichen Bedienung des Programms.
  Der neue bzw. gelegentliche Nutzer sieht – zunächst – mehr den Aufwand, um die Bedienung des Programms zu lernen.
  (ease of use - ease of learning)

Softwareentwicklung

cps4it

Grundsätze für die Softwareentwicklung – 3

- Beide Aussagen haben eine gewisse Berechtigung:
  - Nutzer wissen, was sie wollen!
  - Nutzern muss gesagt werden, was sie wollen!
- Ein Programm, welches der Nutzer nicht nutzt, ist wertlos!
- Der Nutzer weiß, was er will, er kann es nur nicht exakt und nicht vollständig ausdrücken.
- Ein Nutzer kann nicht Dinge wollen, die er nicht kennt!
- 10 % der Wünsche == 90 % des Aufwands.

Softwareentwicklung

cps4it

Grundsätze für die Softwareentwicklung – 4

- Ein Programm sollte so arbeiten, wie der Nutzer es erwartet. Er sollte durch Reaktionen des Programms nicht überrascht werden!
  (no surprises)
- Der Software-Entwickler sollte nicht Probleme lösen, die es nicht gibt!
  Entscheidend sind nicht die Probleme, die der Entwickler sieht, sondern die, die der Nutzer hat!
  Selten ist der Chef der wichtigste Nutzer!
  (Allerdings weiß er dies nicht immer!)

Grundsätze für die Softwareentwicklung – 5

- Folgende Fragen sind abzuwägen:
  - Was kostet es, wenn das Programm einen Fehler zulässt?
  - Was kostet es, ein Programm zu entwickeln, welches keine Fehler zulässt?
  - Was kostet es, ein Programm zu nutzen, welches Fehler zulässt?
  - Was kostet es, wenn ein Programm nicht optimal mit Zeit und Speicherplatz umgeht?
  - Was kostet es, wenn niemand – einschließlich des Entwicklers – in der Lage ist, ein "optimales" Programm (rechtzeitig) zu verändern?

Softwareentwicklung

Grundsätze für die Softwareentwicklung – 6

- Was ein Programmierer nicht in natürlicher Sprache ausdrücken kann, das kann er auch nicht in Programmcode ausdrücken.
- Wenn der Algorithmus zur Lösung eines Problems zu kompliziert wird, dann suche einen neuen. (Mut zum Neuanfang)
- Wenn ein Problem zu umfangreich ist, dann zerlege es! (Teile und herrsche!)

Softwareentwicklung

cps4it

Grundsätze für die Softwareentwicklung – 7

- Ein Problem ist nur dann gut zerlegt, wenn die Teilprobleme wenig voneinander abhängig sind.
- Es ist billiger, einen missratenen Entwicklungsschritt zu wiederholen, als ein missratenes Entwicklungsprodukt jahrelang mühevoll zu warten – oder es nie einzusetzen.

Grundsätze für die Softwareentwicklung – Anmerkungen

- Auch wenn die Grundsätze teilweise lustig wirken, sollte doch der ernste Hintergrund beachtet werden!

- Es wird misslingen, alle Grundsätze gleich wichtig zu nehmen!

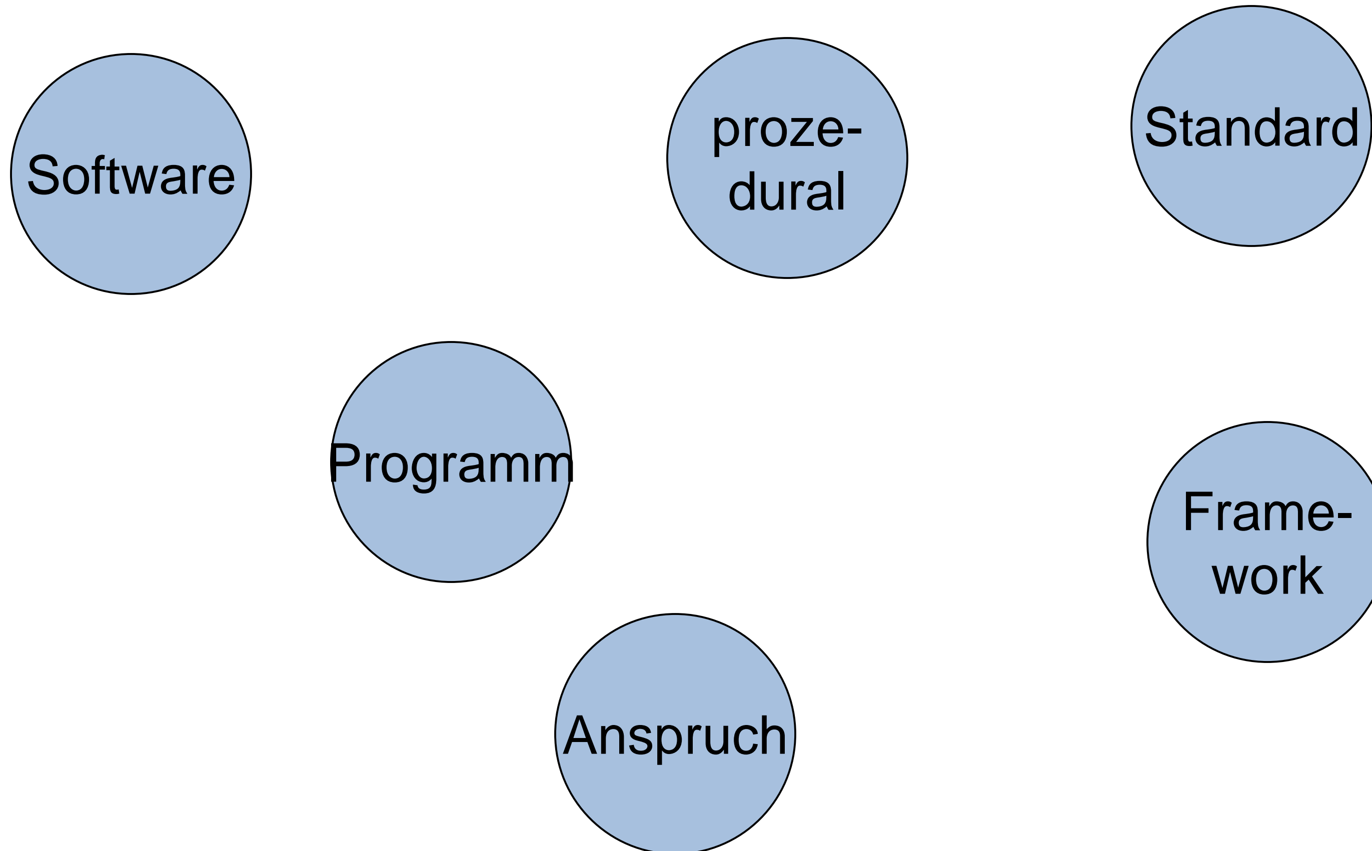
Inhalt

cps4it

- Programmiersprachen
- Softwareentwicklung
- Programmentwicklung
- strukturierte Programmierung
- Abbruchbehandlung

Programmentwicklung

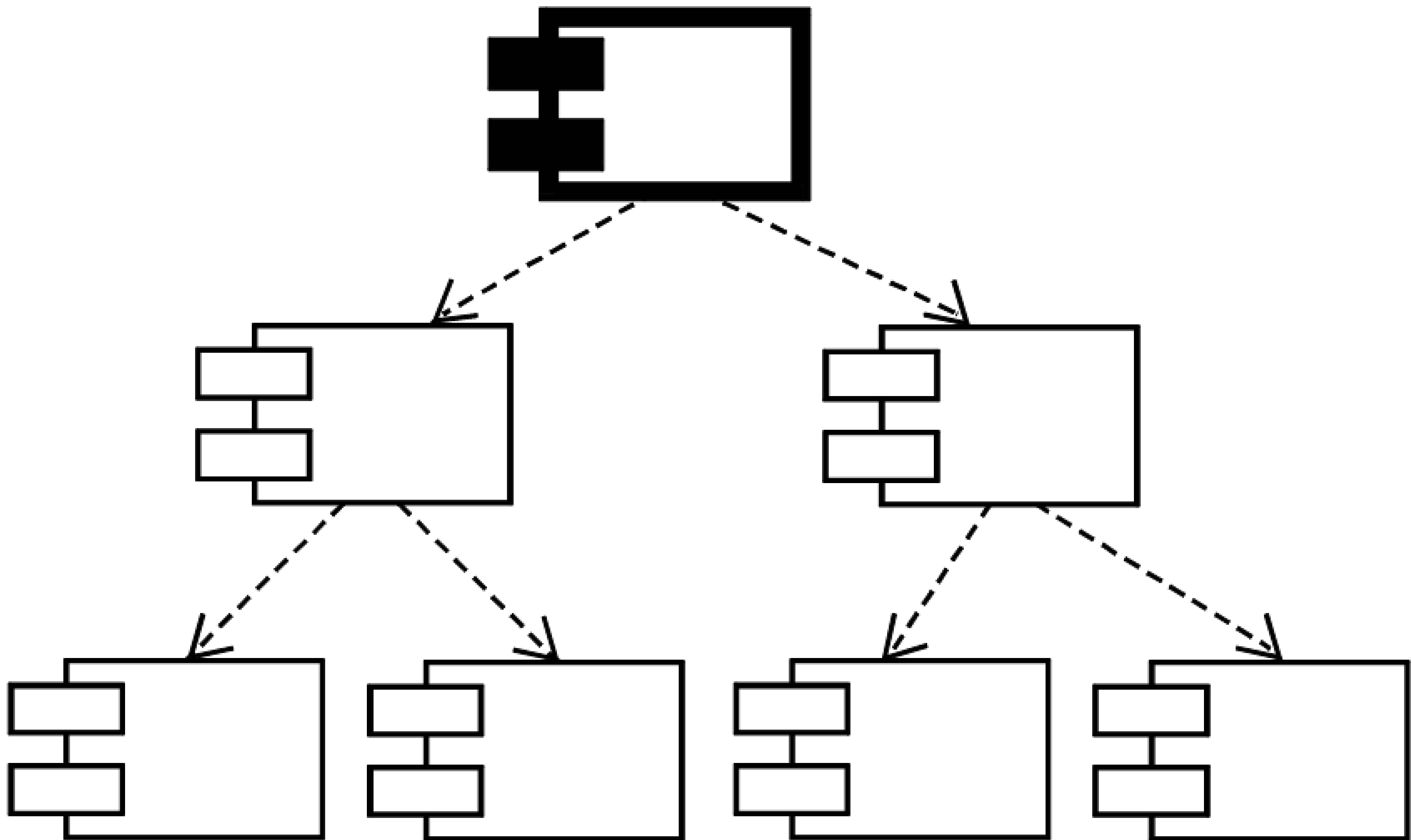
Begriffe



Programmentwicklung

cps4it

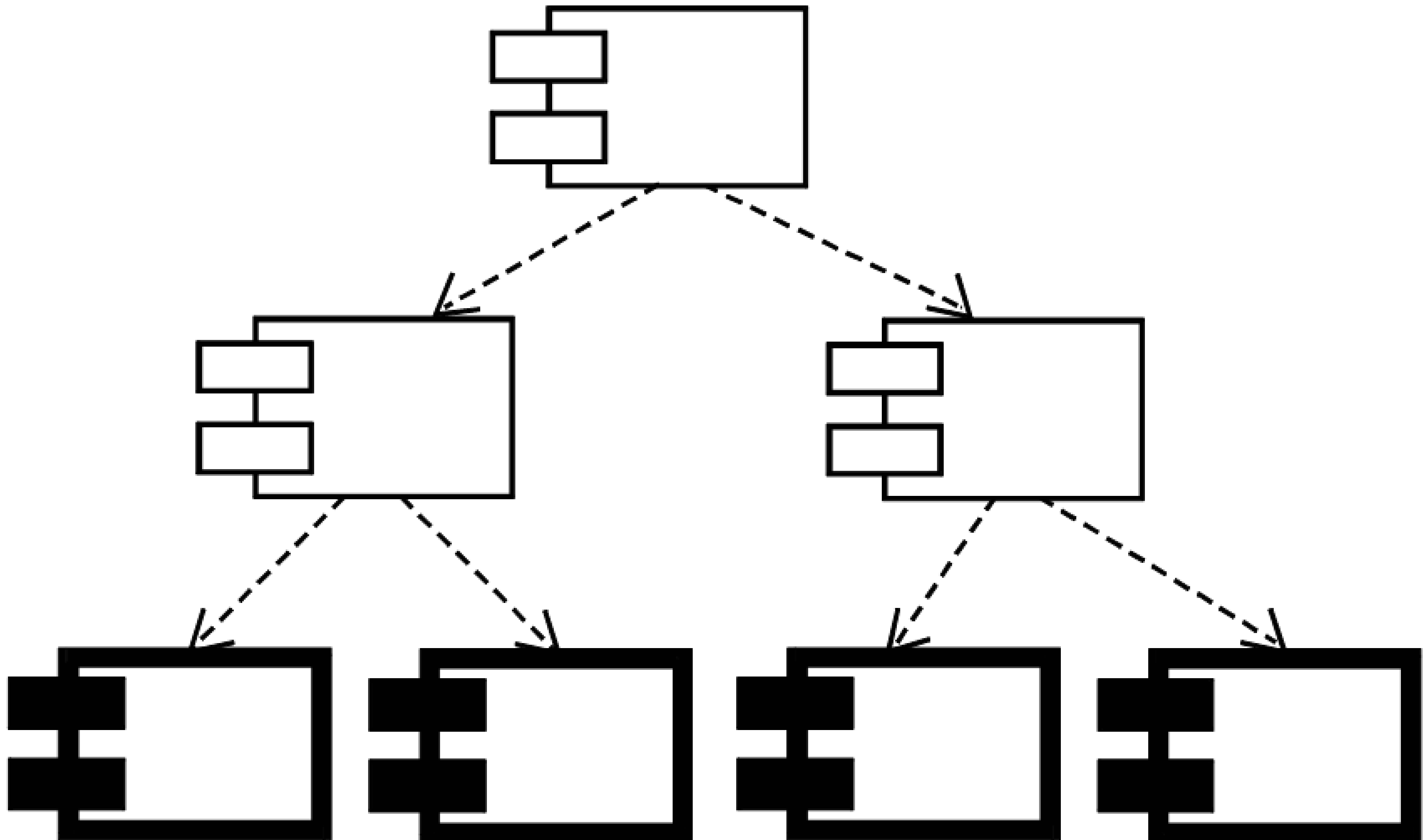
Entwurfstechniken – top-down – vom Ganzen zum Einzelteil



Programmentwicklung

cps4it

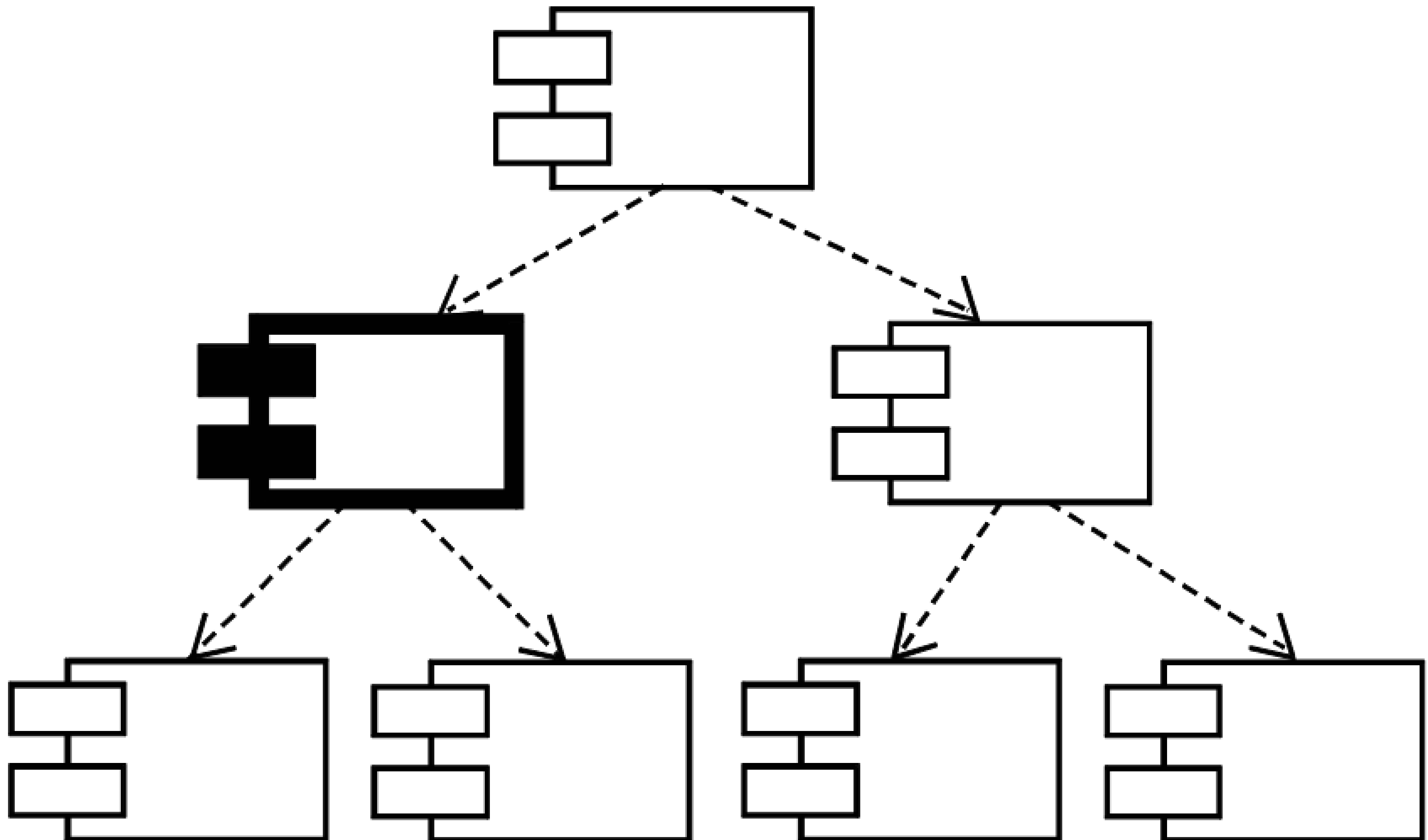
Entwurfstechniken – bottom-up – vom Einzelteil zum Ganzen



Programmentwicklung

cps4it

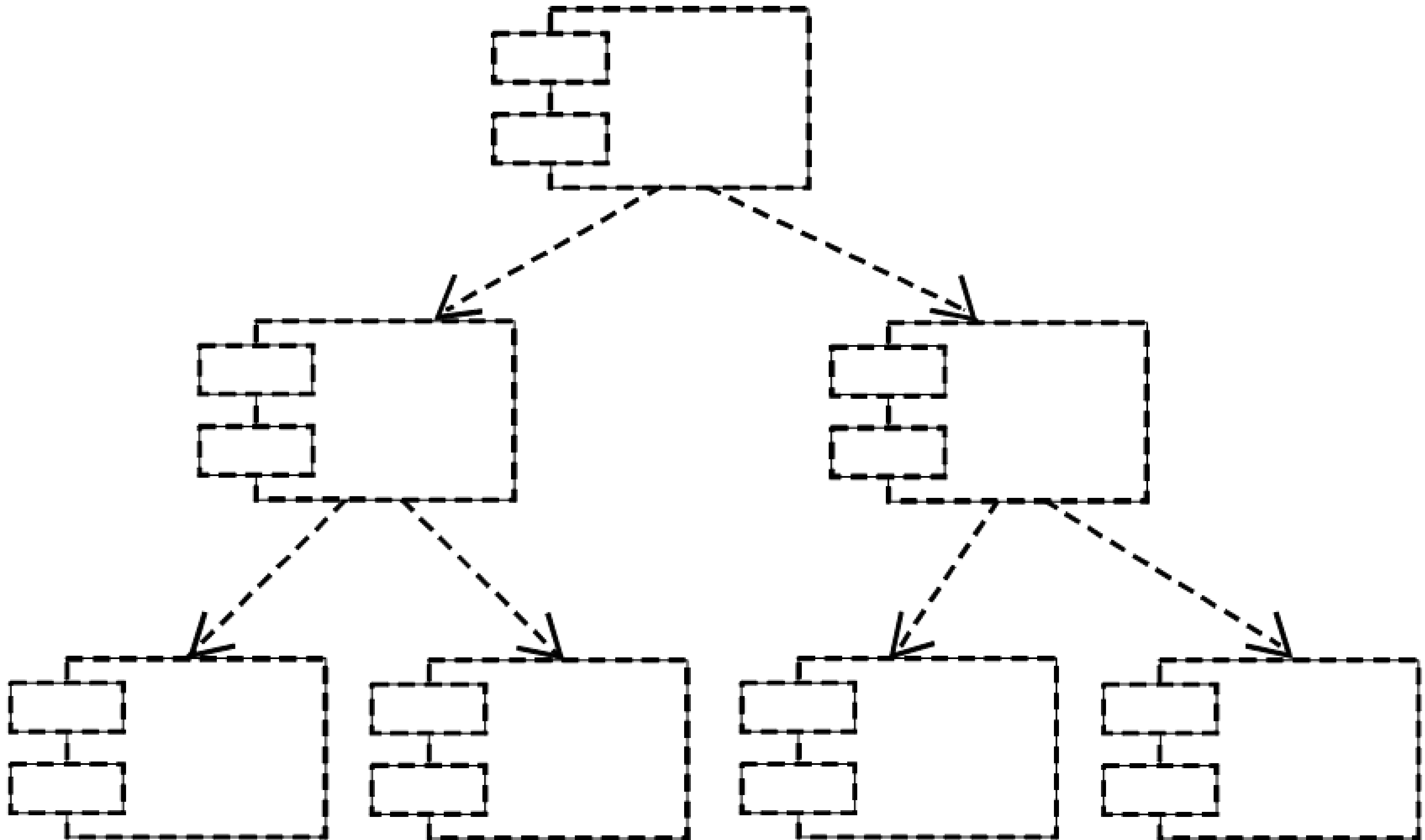
Entwurfstechniken – Hardest first – das Schwierigste zuerst



Programmentwicklung

cps4it

Entwurfstechniken – try and error – Versuch und Irrtum



Programmentwicklung

cps4it

Steuerung des Programmflusses

- deklarativ
  - Beschreibung des Problems
  - Lösung übernimmt der Inferenzmechanismus
- prozedural
  - Algorithmen in sequentieller Reihenfolge
  - kaum beeinflussbar durch außen
- Ereignis gesteuert
  - Beschreibung, wie auf Ereignisse reagiert wird
  - Reihenfolge beliebig

Programmentwicklung

Steuerung des Programmflusses – deklarativ

- Gebunden an Programmiersprache
- keine weite Verbreitung
- hoher Investitionsaufwand zu Beginn

Programmentwicklung

cps4it

Steuerung des Programmflusses – Ereignis gesteuert

- PROC (event,struct_event);
- INIT;
- REPEAT;
  - GetEvent(event);
  - HandleEvent(event);
  - UNTIL quit(event);
- END-REPEAT;
- END;

Programmentwicklung

Steuerung des Programmflusses – prozedural

- PROC;
- INIT;
- action-1;
- action-2;
- ...
- action-n;
- END;

cps4it

Frameworks / Rahmenprogramme – 1

- Rahmen für Programmfluss
- Inhalt wird hinzugefügt
- Vorteile
  - Konzentration auf Lösung
  - allgemeine “Probleme” löst schon der Rahmen
  - Bausteine leichter zu schreiben
  - erprobte Teillösung liegt vor

Programmentwicklung

cps4it

Frameworks / Rahmenprogramme – 2

- Nachteile
  - Effizienz des Programms geringer
- Herausforderungen
  - Existiert ein geeigneter Rahmen?
  - Aufwändige Entwicklung des Rahmens
  - Inhalt und Vorgehensweise des Rahmens muss bekannt sein
- lohnenswert daher erst ab bestimmter Problemgröße -> aber: hat Kolleg/in/e etwas?

Programmentwicklung

cps4it

Standards

- ANSI
  - American National Standards Institute
  - auf Programmiersprachen
  - teilweise auf Umgebung
- ISO
  - International Standards Organization
  - Funktionen innerhalb Programmiersprachen
  - Programmierkonstrukte
- teilweise (freundschaftlich) konkurrierend

Programmentwicklung

Programm – Versuch von Definitionen

- Reihenfolge von Reden, Darbietungen
  - Konferenzprogramm, Konzertprogramm
- Plan, Vorhaben
  - Arbeitsplan
- Grundsätze, Zielstellungen
  - Parteiprogramm
- Aneinanderreihung von Algorithmen
  - Computerprogramm

cps4it

Programm – Gemeinsamkeiten

- Anspruch - es soll etwas erreicht werden
  - Wird das Programm dem Anspruch gerecht?
- Ziel - es gibt eine Vision, ein Ergebnis
  - Wie stehen die Ziele zum Anspruch des Programms?
- Änderungen und Störungen beeinflussen ein Programm
  - Welche Vorkehrungen sind getroffen, damit das Programm keinen Schaden anrichtet?

Programmentwicklung

cps4it

Programm – Unterschiede

- Ein Mensch geht im Allgemeinen sehr flexibel mit Programmen um. Er ist in der Lage, spontan zu reagieren.
- Der Computer nimmt das Programm sehr ernst. Er macht genau und ausschließlich das, was im Programm steht.

Programm – Definitionen

- Fremdwörterbuch, 1977
  - Realisierung eines Algorithmus in der Sprache eines elektronischen Rechenautomaten. Eindeutige und geordnete Zusammenstellung von Befehlen und Daten zur Lösung einer Aufgabe durch elektronische Datenverarbeitungsanlagen.
- Schülerduden, 1986
  - Formulierung eines Algorithmus und der dazugehörigen Datenbereiche in einer Programmiersprache
- Webster's 1995
  - a sequence of coded instructions for a computer

Programmentwicklung

cps4it

Programm – einfaches Schaubild

Programmentwicklung

cps4it

Auswahl der Programmiersprache . . . – 1

- ... der/die Entwickler beherrschen
- ... dem Entwickler am besten gefällt
- ... dem Entwicklerteam am besten gefällt
- ... dem Chef am besten gefällt
- ... dem Kunden am besten gefällt
- ... von vielen anderen in vergleichbaren Fällen verwendet wird
- ... sehr schnellen Programmcode erwarten lässt
- ... sehr Speicher sparenden Programmcode erwarten lässt

Programmentwicklung

cps4it

Auswahl der Programmiersprache . . . – 2

- ... sehr gut lesbaren Quelltext erwarten lässt
- ... es dem Programmierer schwer macht, unbemerkt Programmierfehler in ein Programm einzubauen
- ... auf allen wichtigen Rechnerplattformen zur Verfügung steht ...
- ... und in Zukunft auf allen wichtigen Rechnerplattformen zur Verfügung stehen wird
- ... für die es zukünftig noch Entwickler geben wird

Programmentwicklung

technische Kriterien für die Auswahl der Programmiersprache

- Namen - Länge
- Konstanten und Typen
  – Strukturen, Typprüfung, symbolische Variablen
- Wertzuweisungen
  – eindeutig ohne Nebenwirkungen
- Ablauf und Stil
  – Konstrukte klar, Wesen klar und deutlich, nicht zusammen gestoppelt
- Compiler
  – verfügbar, effizienter Code, benutzerfreundlich
- Portabilität wenn erforderlich

Programmentwicklung

cps4it

weitere Kriterien für die Auswahl der Programmiersprache ;-)

- ADA — Ein als amerikanischer Straßenkreuzer getarnter Schützenpanzerwagen.
- Assembler — Ein Go-Cart ohne Sicherheitsgurt und Überrollbügel. Gewinnt jedes Rennen, wenn es nicht vorher im Graben landet.
- BASIC — Eine Ente - weder modern noch besonders schnell, aber für jeden erschwinglich. Und manch einer, der sich daran gewöhnt hat, will gar nichts anderes mehr haben.
- C — Ein offener Geländewagen. Kommt durch jeden Matsch und Schlamm, der Fahrer sieht hinterher auch entsprechend aus.
- COBOL — Ein dunkelroter Benz mit getöntem Panzerglas und kostbaren Intarsienarbeiten im Fond. Kein Mensch fährt diesen Wagen selbst; man läßt fahren.
- FORTRAN — Ein Schlitten aus den fünfziger Jahren mit riesigen Heckflossen. Erntet bei der technischen Überprüfung stets mißtrauische Blicke, überholt aber noch manch neueres Gefährt.
- LISP — Ein Prototyp mit Telepathie-Steuerung. Kann außer von seinem Erfinder von niemanden bedient werden.
- PASCAL — Entwurf eines amerikanischen Straßenkreuzers, der nur durch Versehen in die Serienproduktion gelangte.
- MODULA-2 — Wie Pascal, aber mit dreifachen Sicherheitsgurten, seitlichen Stoßstangen und separatem Gaspedal für jeden der fünf Gänge.
- PL/1 — Ein handgefertigter Eigenbau, mit Teilen von FORTRAN, COBOL, PASCAL und ADA. Entsprechend sieht er aus.
- PROLOG — Enthält statt eines Lenkrades eine Automatik, die alle Straßen solange absucht, bis das gewünschte Fahrziel erreicht ist.

Programmentwicklung

cps4it

weitere Kriterien für die Auswahl der Programmiersprache ;-)

- Fluchen ist die einzige Sprache, die alle Programmierer wirklich beherrschen.
- Man muß ein Idiot sein, um das Programm eines anderen Idioten verstehen zu können.
- Die einzige gute Programmiersprache ist die, die man nicht benutzen darf.
- Es hat wenig Sinn, eine Programmiersprache verstehen zu wollen.

Programmentwicklung

cps4it

wichtige Aussagen zu Programmen ;-)

- Jedes fertige Programm, das läuft, ist veraltet.
- Wenn ein Programm korrekt läuft, muss es geändert werden.
- Wenn ein Programm nutzlos ist, muss es dokumentiert werden.
- Erst wenn ein Programm mindestens 6 Monate läuft, wird der schlimme Fehler entdeckt werden.
- Der Fehler befindet sich immer in der Routine, die niemals getestet wurde.
- Es gibt immer einen Fehler mehr als gedacht.

Programmentwicklung

cps4it

und … ;-)

- Ein Computerprogramm tut, was du schreibst, nicht was Du willst.
- Es ist unmöglich, ein Programm fehlerfrei zu programmieren; Dummköpfe sind erfinderisch.

- Bedenke: keine Sprache an sich ist gut oder schlecht, sondern der, der sie benutzt!
  – Wenn man einen Teelöffel Wein in ein Fass Jauche gießt, ist das Resultat Jauche.
  – Wenn man einen Teelöffel Jauche in ein Fass Wein gießt, ist das Resultat ebenfalls Jauche.

Inhalt

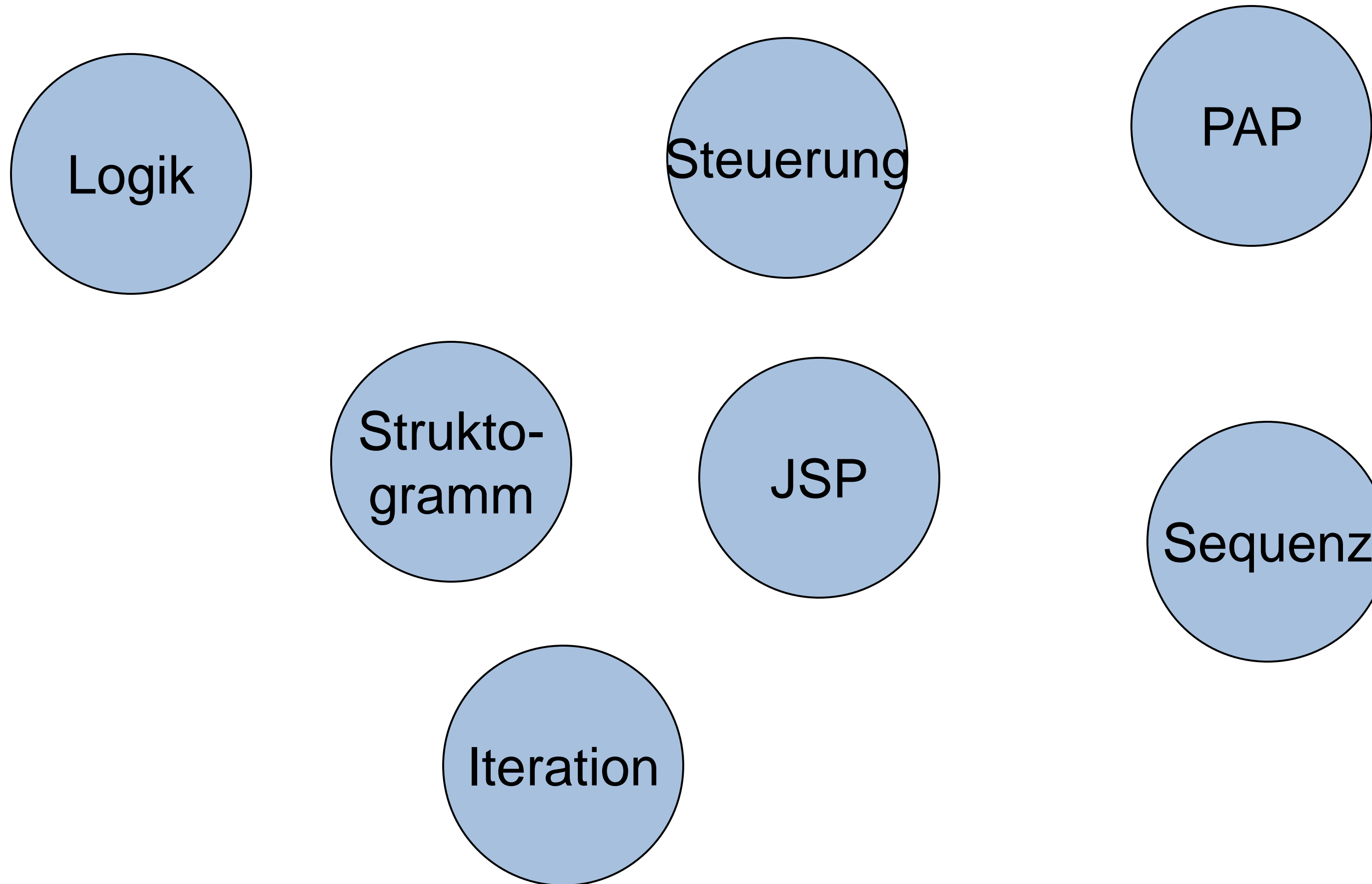
cps4it

- Programmiersprachen
- Softwareentwicklung
- Programmentwicklung
- strukturierte Programmierung
- Abbruchbehandlung

strukturierte Programmierung

cps4it

Begriffe



strukturierte Programmierung

cps4it

Grundkonzepte

- Bildung von logischen Programmeinheiten
- hierarchische Programmorganisation
- Definition einer zentralen Programmsteuerung
- Beschränkung der Ablaufsteuerung
- Beschränkung der Datenverfügbarkeit

strukturierte Programmierung

cps4it

Grundelemente

- Sequenz
- Verzweigung
  - (un)vollständige Alternative
  - Mehrfachverzweigung
  - Fallauswahl
- Iteration
  - (Nicht)Abweisschleife
  - verallgemeinerter Zyklus

strukturierte Programmierung

cps4it

Darstellungsmittel

- Struktogramm
- JSP-Diagramm
- Programmablaufplan
- Pseudocode
- “höhere Programmiersprache”

strukturierte Programmierung

cps4it

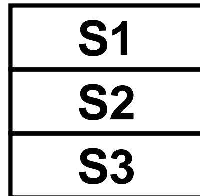
Vorgehensweise

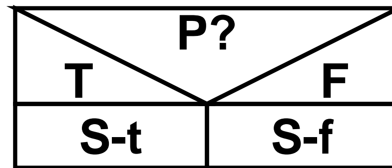
- Fluss zurückführen auf Grundelemente
- von oben nach unten (konsequent)
- komplexe Aktionen schrittweise verfeinern
- Grundelement kann weitere Grundelemente einschließen
- Unterprogramme werden separat behandelt
- Komplexität vermindern
  - der Mensch kann nicht mehr als 5 bis 9 verschiedene Informationen gleichzeitig erfassen

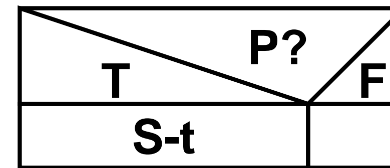
strukturierte Programmierung

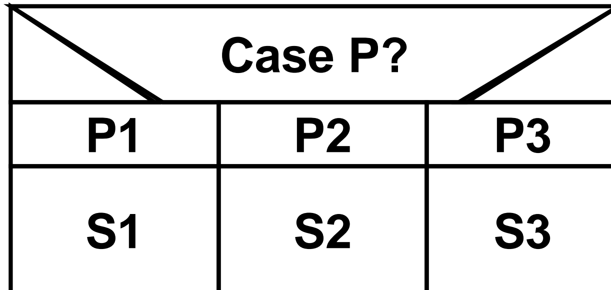
cps4it

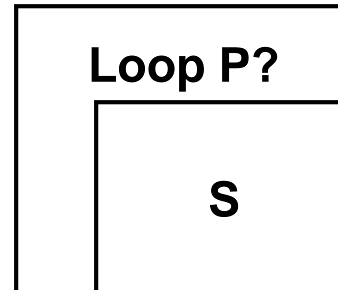
Struktogramm – Grundelemente

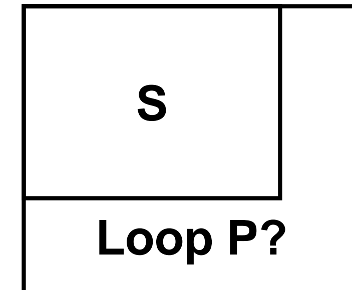








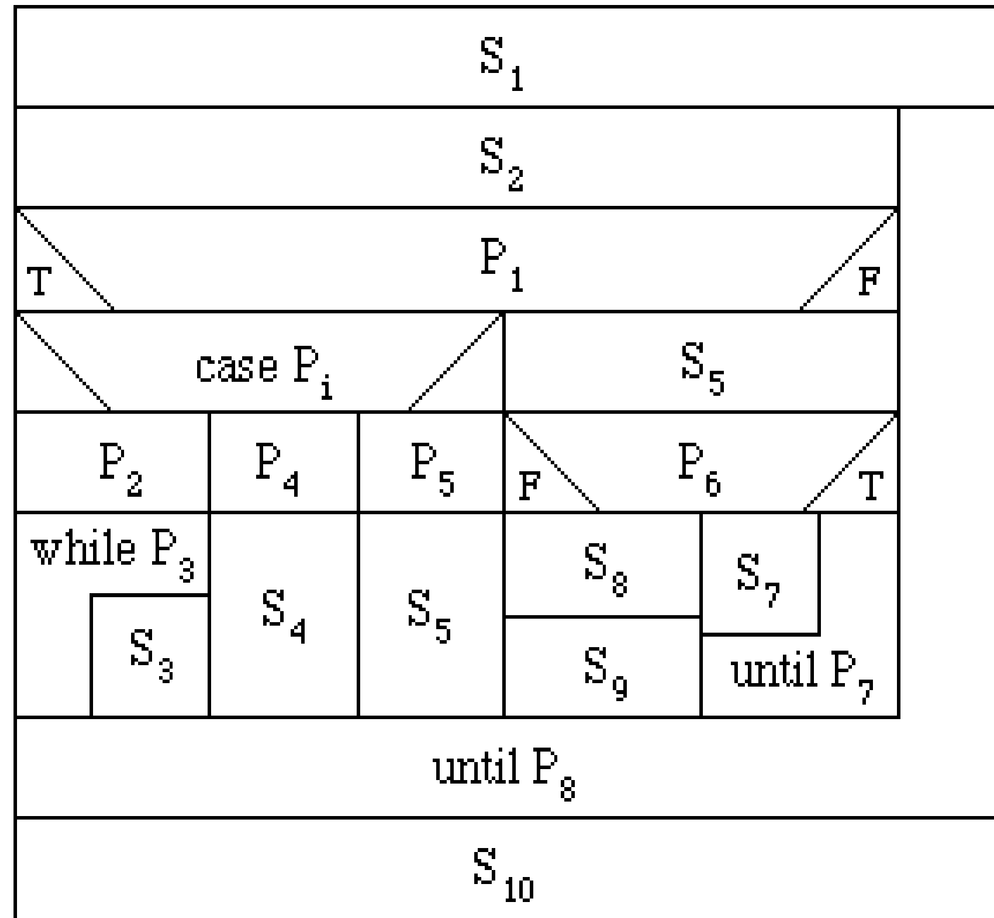




strukturierte Programmierung

Struktogramm – Beispiel

```
S1;
repeat
   S2
   if P1 then
      case (Pi | i = 2,4,5)
      of P2:
         while P3 do S3 end while
      of P4:  S4
      of P5:  S5
      end case
   else S6;
      if P6 then
         repeat S7 until P7
      else  S8; S9
      end if
   end if
until P8;
S10.
```

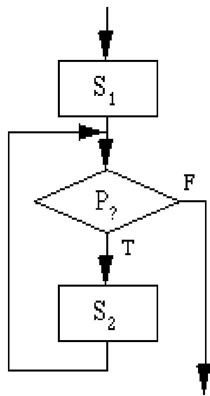


Struktogramm – Möglichkeiten und Grenzen

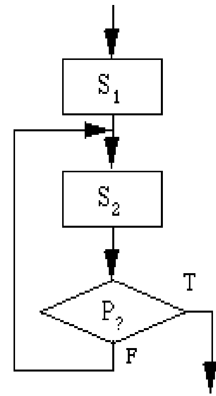
- Maximal 1 Seite pro Struktogramm
- Modularisieren
- es gibt unterstützende Software
  - Weiterentwicklung auf Basis Struktogramm möglich
  - EasyCase (Siemens)
  - AllFusion:Gen (CA)
  - etc.

strukturierte Programmierung

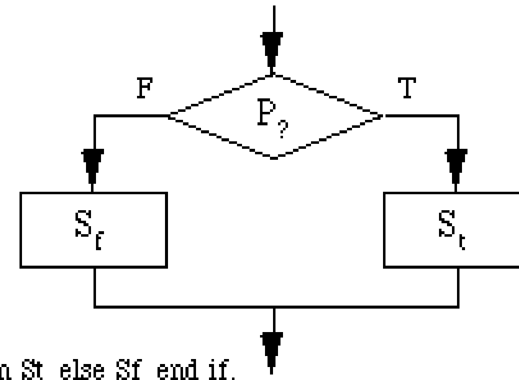
Programmablaufplan

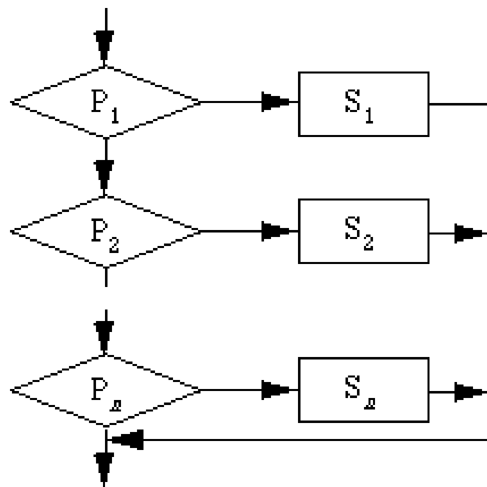


while P do S2 end while.

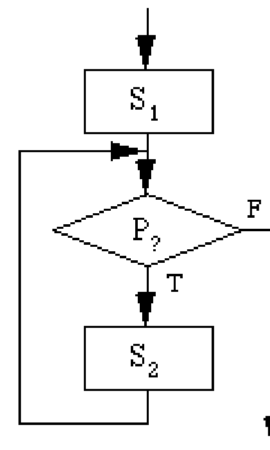


repeat S2 until P end repeat.

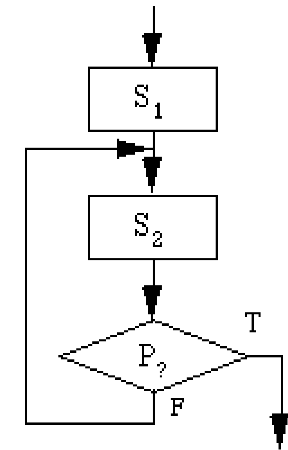




case P?
of P1: S1;
of P2: S2;
...
of P_n: S_n;
end case.



while P do S2 end while.



repeat S2 until P end repeat.

strukturierte Programmierung

JSP – Jackson structured programming

sequence
seq
this
that
other

selection
sel
o c1
this
o c2
that
o else
other

iteration
plural
w c1
singular

subprogram call
subprog

subprogram call
subprog

recursive subprogram call
subprog

indirect subprogram call

in only data flow
subprog

out only data flow
subprog

in out data flow
subprog

Optional In Only Data Flow
subprog

Terminal or File Input
subprog

quit
! q1

posit
? p1

admit
? a1

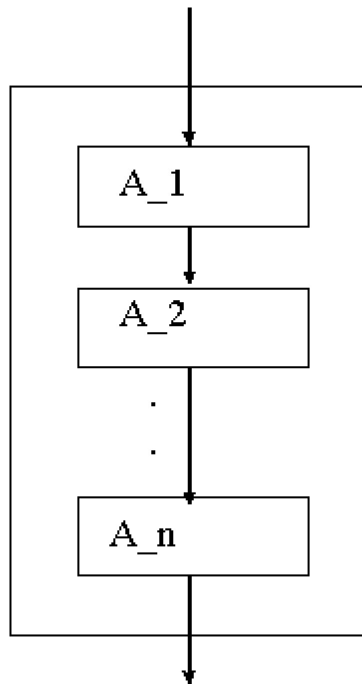
Subprogram Parameters
subprog()
subprog

Generic Units
Subprogram
or

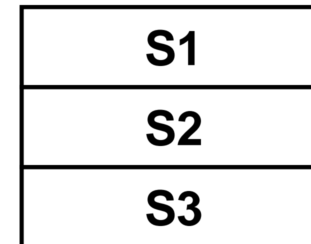
strukturierte Programmierung

Sequenz

Anweisungssequenz




```
begin
   A_1;
   A_2;
   ....
   A_n;
end;
```



strukturierte Programmierung

Sequenz

S1
S2
S3

- ist eine Folge von Anweisungen
  - Wertzuweisungen
  - Aufrufe von Unterprogrammen, Prozeduren, Funktionen
  - Sprungbefehle
  - Alternativen
  - Zyklen
- bei Fehler kann die Sequenz unterbrochen werden

strukturierte Programmierung

cps4it

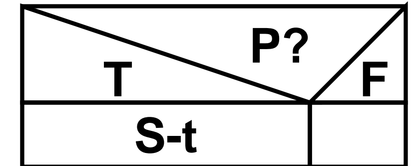
Sequenz – Beispiele

S1
S2
S3

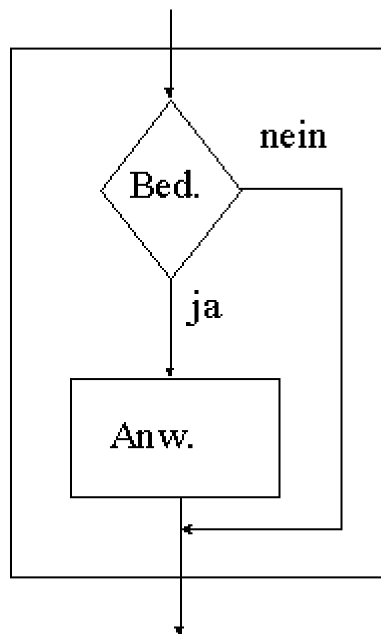
MOVE A TO B	B = A;	b = a;
CALL HUGO	CALL HUGO;	call hugo;
GO TO OTTO	GO TO OTTO;	goto otto;
IF / ELSE	IF / ELSE	if / else
PERFORM	DO	loop while
CONTINUE	;	say
END	END;	end
GOBACK	RETURN;	return

strukturierte Programmierung

unvollständige Verzweigung



Einseitige Auswahl

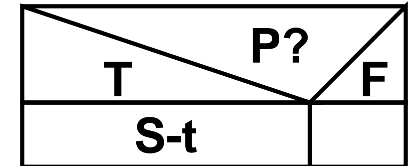


if Bedingung
then Anw ;

- S-t ist eine Folge von beliebigen Anweisungen
- S-t wird genau dann ausgeführt, wenn P erfüllt ist

strukturierte Programmierung

unvollständige Verzweigung – Beispiel

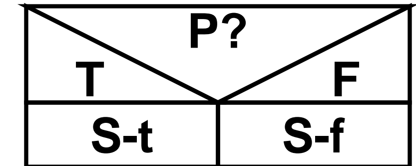



```
if environment = 'TSO' then do;
   say 'Umgebung = ' sysvar(sysenv);
end;
```

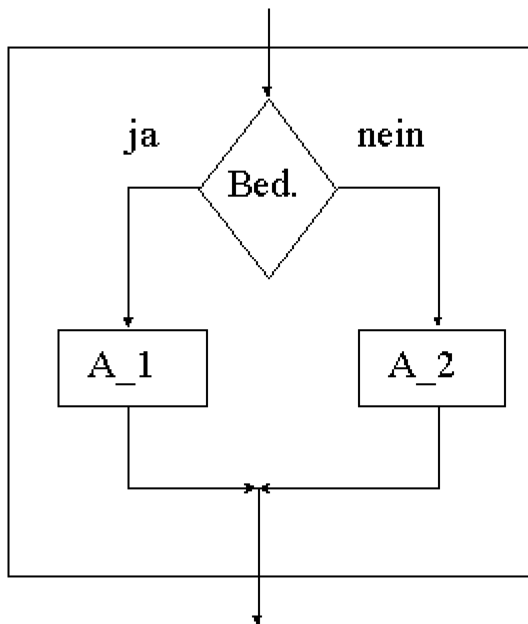
strukturierte Programmierung

cps4it

vollständige Verzweigung



Zweiseitige Auswahl

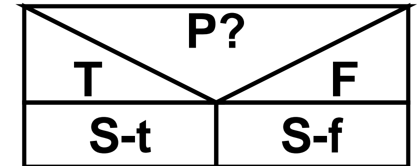



```
if Bedingung
    then A_1
    else A_2;
```

- S-t und S-f sind Folgen von beliebigen Anweisungen
- S-t wird genau dann ausgeführt, wenn P erfüllt ist
- S-f wird genau dann ausgeführt, wenn P nicht erfüllt ist

strukturierte Programmierung

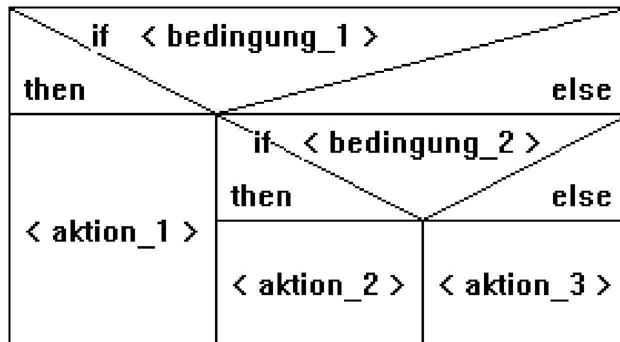
vollständige Verzweigung – Beispiel



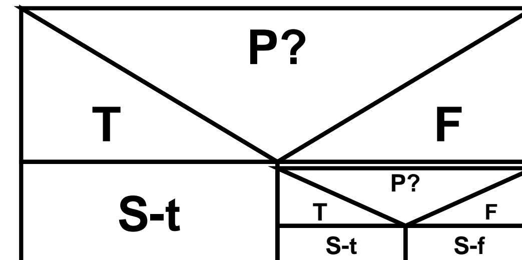

```
if environment = 'TSO‘
then do;
   say 'Umgebung = ' sysvar(sysenv);
end;
else do;
   say 'wo bin ich denn?’
end;
```

strukturierte Programmierung

Mehrfachverzweigung



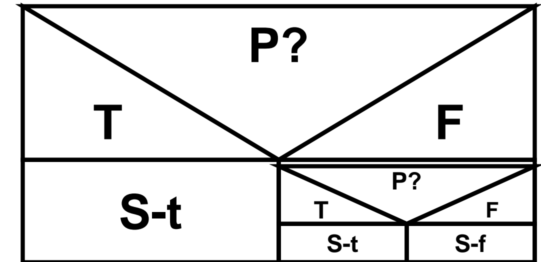
Kein separates PAP-Bild



- Die alternativ auszuführenden Aktionen hängen von verschiedenen Bedingungen ab, die in einer vorgegebenen Reihenfolge ausgewertet werden, d.h. die Bedingungen beruhen auf der Auswertung unterschiedlicher Ausdrücke.

strukturierte Programmierung

Mehrfachverzweigung – Beispiel



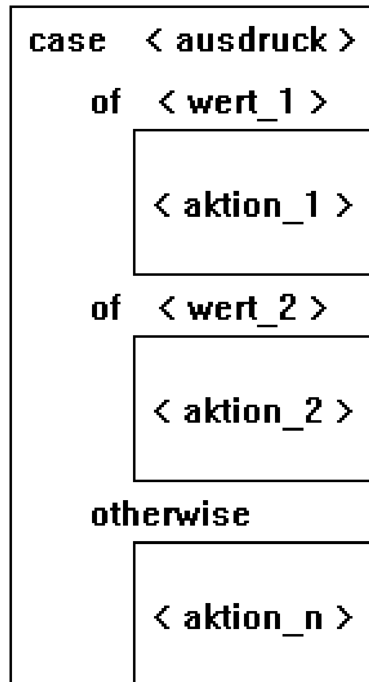

```
if enviroment = 'TSO' then do;
   say 'Umgebung = ' sysvar(sysenv);
end;
else do;
   if environment = ’CICS’ then do;
      say 'Umgebung = CICS’;
   end;
   else do;
      say 'wo bin ich denn?’
   end;
end;
```

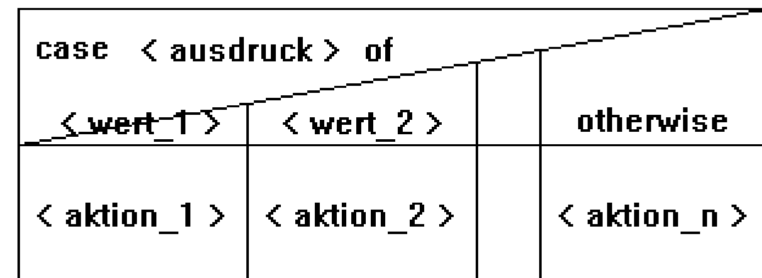
Fallauswahl

- Die alternativ auszuführenden Aktionen hängen von der Auswertung eines einzigen Ausdrucks ab. Kriterium ist der Wert, den dieser Ausdruck liefert.
- Die Fallauswahl ist eine spezielle Form der Mehrfachverzweigung. Die für sie in vielen Programmiersprachen vorgesehene spezielle Notationsform ist jedoch einfacher und übersichtlicher: Alle Bedingungen der Mehrfachverzweigung lassen sich auf die Auswertung nur eines Ausdrucks zurückführen.

strukturierte Programmierung

Fallauswahl – Struktogramm

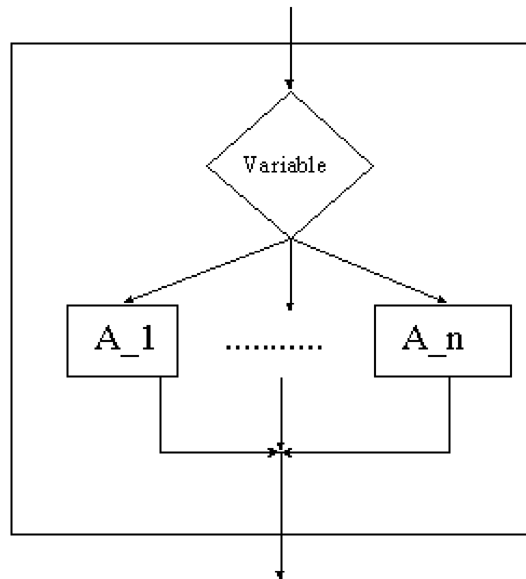




strukturierte Programmierung

Fallauswahl – Programmablaufplan

Mehrfachauswahl

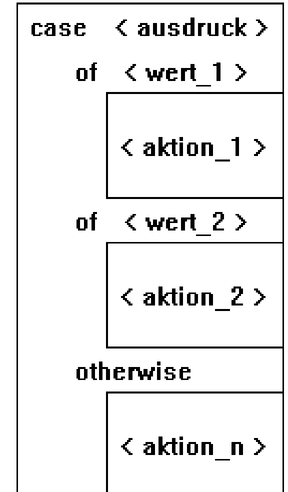



```
CASE Ausdruck of
      wert_1: A_1;
      wert_2: A_2;
         .
         .
         .
      wert_n: A_n;
END; { case }
```

strukturierte Programmierung

Fallauswahl – Beispiel

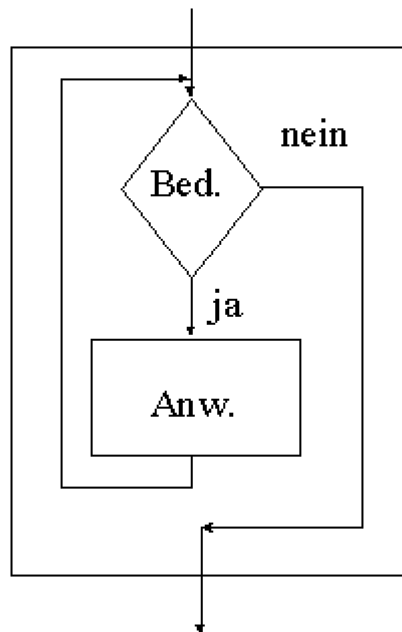
```
select (environment);
   when ('TSO’)
      say ’Ich bin im TSO’;
   when (’CICS’)
      say ’Ich bin im CICS’;
   otherwise
      say ’Ach du Schrott!!’;
end-select;
```



strukturierte Programmierung

Abweisschleife

Abweisende Schleife



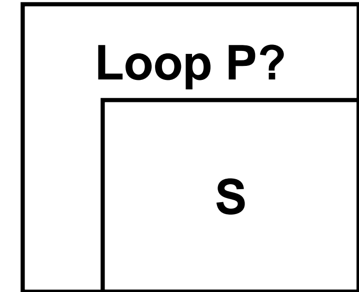
while Bedingung do Anw;

der Schleifenkörper kann 0-mal ausgeführt werden, wenn die Bedingung schon zu Beginn *falsch* ist.

strukturierte Programmierung

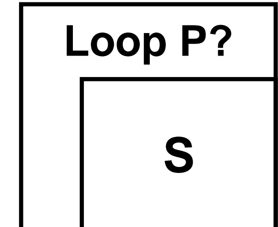
Abweisschleife

- Die Aktion wird solange wiederholt, wie die Bedingung erfüllt ist.
- Die Bedingung wird vor der Aktion geprüft, d.h. die Aktion wird möglicherweise nie ausgeführt.



strukturierte Programmierung

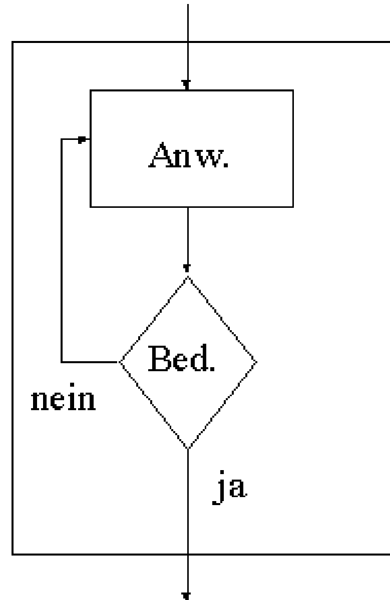
Abweisschleife – Beispiel




```
do while (buli1="bayern")
   say "Auf! schalke, mach mal!"
   call bundesligaspiele
   if punkte(schalke) > punkte(bayern)
      buli1 = "schalke"
   end-if
end-do
```

Nichtabweisschleife

Nicht abweisende Schleife

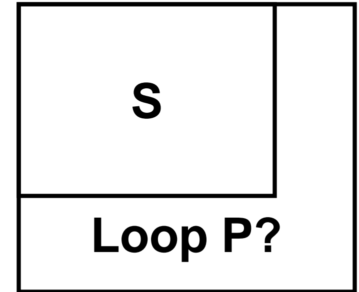



```
repeat
   A_1;
   A_2;
    .
    .
   A_n;
until Bedingung;
```

der Schleifenkörper wird mindestens 1-mal ausgeführt, auch wenn die Bedingung schon zu Beginn *falsch* ist.

strukturierte Programmierung

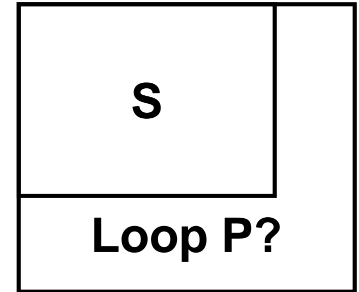
Nichtabweisschleife



- Die Aktion wird solange wiederholt, bis die Bedingung erfüllt ist.
- Sie wird immer mindestens einmal ausgeführt.

strukturierte Programmierung

Nichtabweisschleife – Beispiel




```
do until (buli1="schalke")
   say "Auf! schalke, mach mal!"
   call bundesligaspiele
   if punkte(schalke) > punkte(bayern)
      buli1 = "schalke"
   end-if
end-do
```

strukturierte Programmierung

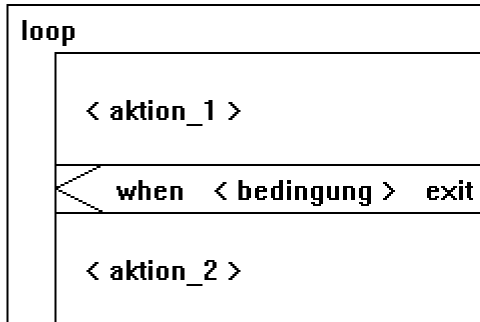
cps4it

Abweisschleife / Nichtabweisschleife

- Achtung!
  In den Programmiersprachen sehr uneinheitliche Sprachregelung.
  Genau hinsehen, was der Loop-Befehl bewirkt.

strukturierte Programmierung

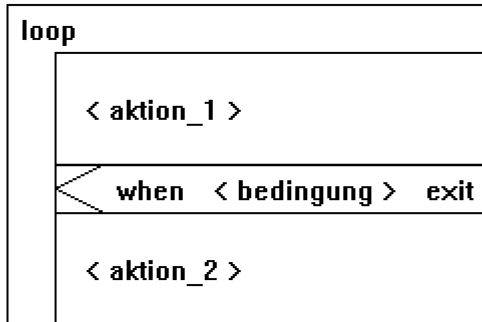
verallgemeinerter Zyklus



- Innerhalb des Aktionsteils befindet sich eine Abbruchbedingung, d.h. der Zyklus wird verlassen, wenn diese Bedingung erfüllt ist.
- Die Aktionen 1 und 2 werden wiederholt, bis die Abbruchbedingung erfüllt ist. Aktion 1 wird dabei einmal mehr ausgeführt als die Aktion 2.
- Fast keine Sprache kennt diesen Konstrukt.

strukturierte Programmierung

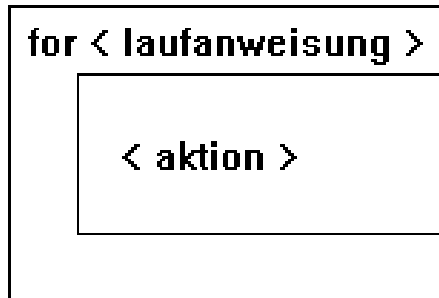
verallgemeinerter Zyklus – Beispiel




```
do for ever;
   say "Auf! schalke, mach mal!"
   call bundesligaspiele
   if punkte(schalke) > punkte(bayern)
      leave
   end-if
end-do
```

strukturierte Programmierung

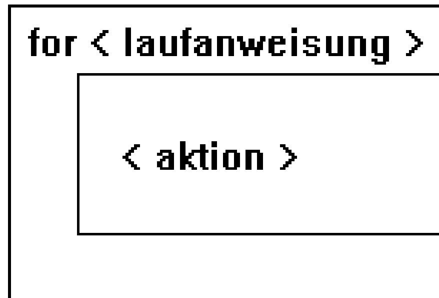
Zählschleife



- Es gibt eine Laufvariable, die von einem festen Anfangswert in einer Schrittweite bis zu einem Endewert verändert wird.
- Pro Wert der Laufvariable wird aktion ausgeführt.

strukturierte Programmierung

Zählschleife – Beispiel




```
do i = 1 to CPUS.0
  say 'CPU' i ' has CPU info ' CPUS.i
end
```

strukturierte Programmierung

cps4it

Rekursion

- besondere Art einer Schleife
- kein eigenes Struktogramm definiert
- Beispiel: Fakultät
  - n! = n * (n-1)! für n > 1
  - n! = 1 für n = 1
- wichtig sind
  - Rekursionsvorschrift (was ist zu tun)
  - Rekursionsverankerung (wie lange ist es zu tun)

strukturierte Programmierung

cps4it

Sprungbefehl

- Ein Sprungbefehl widerspricht den Regeln der strukturierten Programmierung.
- Typen
  - direkte Sprünge zu (fast) beliebigem Code
  - kontrollierte Sprünge in Form von Verlassen einer Sequenz

strukturierte Programmierung

Zusicherungen

```
   Vorbedingungen           Nachbedingungen
                |                |
                V                V
                ==================
  Eingangs-     |   Programm-    |  Ausgangs-
------------->|                |------------->
   größen       |   abschnitt    |   größen
                ==================
                :                :
                :..Invarianten..:
```

Inhalt

cps4it

- Programmiersprachen
- Softwareentwicklung
- Programmentwicklung
- strukturierte Programmierung
- Abbruchbehandlung

Abbruchbehandlung

cps4it

Begriffe

ABEND

Abbruch

Speicher

Datei

Strukto-
gramm

Anwei-
sung

Abbruchbehandlung

cps4it

Situationen

- Hardware-Probleme
- Software-Probleme
- Kommunikationsprobleme
- Autorisierungsprobleme
- Verfahrensprobleme
- Datenprobleme

Abbruchbehandlung

cps4it

Auslöser und Behandlung

- Auslöser im Normalfall
  - Technik
  - technisches Modul
  - selten Konstrukt in der Programmiersprache
- Behandlung
  - durch Technik
  - durch technisches Modul
  - selten Konstrukt in der Programmiersprache
  - keinen erneuten Fehler provozieren!

Abbruchbehandlung

must nach Ausnahmebehandlung

- das Schließen von Dateien
- das Aufheben von Zugriffssperren
- die Freigabe von dynamisch angefordertem Speicherplatz
- also: gesamtes technisches Umfeld sauber zurücksetzen für
  - Neustart
  - Restart

Inhalt

cps4it

- Programmiersprachen
- Softwareentwicklung
- Programmentwicklung
- strukturierte Programmierung
- Abbruchbehandlung